Pôle Systèmes et Réseaux (C3)

S3.A.01 – Développement d'une application

Tuteur : Pr. Richard Chbeir

Projet : Hego Lagunak, une application de parrainage pour le BDE

Équipe 3 : BRIERRE Titouan (TP1), DARGAZANLI Nicolas (TP1), ERREZARET Leho (TP2) et MAURICE Alexandre (TP1), en BUT Informatique, 2022, Semestre 3, Parcours A.

Table des matières

1er jalon : analyse

1. Expression des besoins

Dans le cadre de cette SAE nous devons mettre en place toute une infrastructure réseau permettant la mise en service de notre application.

Notre application doit pouvoir afficher des pages web, réaliser des requêtes entre les clients et l'application, appliquer des traitements à des informations relatives aux clients, répondre à des contraintes de sécurité et éventuellement disposer de sauvegardes en cas de problème.

2. Les différents services

Un serveur Apache / NGINX : serveurs Web destinés à afficher les pages, avec PHP installé pour exécuter les scripts PHP.

Un serveur MariaDB / MySQL / Oracle : serveur de base de données pour stocker les informations nécessaires (utilisateurs, formulaires etc.).

Un pare-feu : pour faire respecter la politique de sécurité du réseau.

Un répartiteur de charge : éventuellement, pour rendre le traitement global plus efficace.

Un serveur de clonage de données : pour sauvegarder les données utilisées par l'application.

3. Les machines

Chaque service aura une machine dédiée, mais nous devons prendre en compte l'existence une machine administrateur et une machine client (à l'extérieur). Faire ce choix nous permet d'avoir plus de sécurité, sans « planter » toute l'application si une machine ne fonctionne plus. De plus cela empêchera certains serveurs d'avoir un accès vers le réseau internet (en particulier le SGBD).

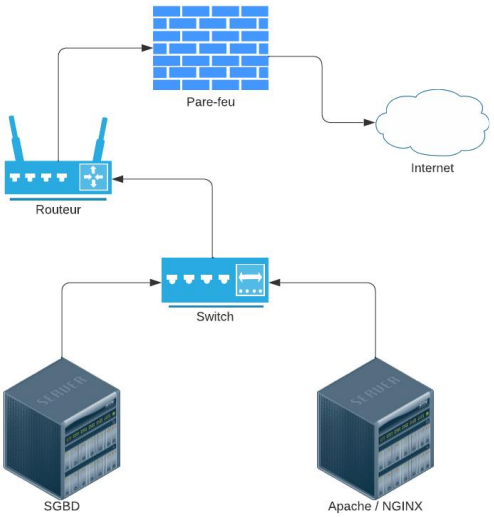
4. Réseaux

Nous n'aurons besoin que d'un seul réseau qui comprendra l'ensemble des serveurs avec une seule passerelle vers le monde extérieur. Nous ne considérons pas les machines de test client comme appartenant à notre réseau.

En vue du nombre maximum d'utilisateurs (environ 200 utilisateurs avec une fréquence d'utilisation basse sur une période de 1 mois), nous ne nous servirons probablement pas d'un répartiteur de charge.

5. Plan d’adressage

Nous avons choisi un sous-réseau de classe C : 192.168.27.0/24.



2ème jalon : création de l'infrastructure réseau

Les clients

Nous devons paramétrer les deux clients pour qu'ils récupèrent leur adresse IP via le protocole DHCP.

Pour cela, nous ajoutons le paramétrage de l'interface eth0, comme nous le montrons sur la capture ci-dessous.

Le serveur DHCP

Le serveur DHCP permettra de desservir des adresses IP ainsi que le routage par défaut.

Nous définissons de manière locale les adresses IP.

eth0 : 192.168.0.252/24

eth1 : 192.168.1.252/24

Nous paramétrons le service DHCP pour qu'il écoute sur l'interface eth0, grâce à la ligne `INTERFACES="eth0"`.

```
# Defaults for isc-dhcp-server initscript
# sourced by /etc/init.d/isc-dhcp-server
# installed at /etc/default/isc-dhcp-server by the maintainer scripts

#
# This is a POSIX shell fragment
#

# Path to dhcpd's config file (default: /etc/dhcp/dhcpd.conf).
#DHCPD_CONF=/etc/dhcp/dhcpd.conf

# Path to dhcpd's PID file (default: /var/run/dhcpd.pid).
#DHCPD_PID=/var/run/dhcpd.pid

# Additional options to start dhcpd with.
#       Don't use options -cf or -pf here; use DHCPD_CONF/ DHCPD_PID instead
#OPTIONS=""

# On what interfaces should the DHCP server (dhcpd) serve DHCP requests?
#       Separate multiple interfaces with spaces, e.g. "eth0 eth1".
INTERFACES="eth0"
```

Nous allons maintenant configurer le réseau que le serveur DHCP va redistribuer.

```
  GNU nano 2.2.6          File: /etc/dhcp/dhcpd.conf


subnet 192.168.0.0 netmask 255.255.255.0 {
  range 192.168.0.1 192.168.0.250;
  option routers 192.168.0.254;
  option broadcast-address 192.168.0.255;
  default-lease-time 600;
  max-lease-time 7200;
}
```

Les routeurs

Routeur1 :

On attribue les dernières IP des réseaux disponibles et nous pointons le Routeur2 comme routeur par défaut.

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

ifconfig eth1 192.168.1.254/24
ifconfig eth0 192.168.0.254/24
route add default gw 192.168.1.1
echo 1 > /proc/sys/net/ipv4/ip_forward
exit 0
```

Routeur2 :

On attribue les premières IP des réseaux disponibles et nous pointons la passerelle comme routeur par défaut. Etant donné que le Routeur2 est le routeur par défaut des serveurs, il faut indiquer le routage vers le réseau LAN1, ce qui engendre une redirection de Routeur2 vers Routeur1.

```
 GNU nano 2.2.6            File: /etc/rc.local

#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.
ifconfig eth0 192.168.1.1/24
ifconfig eth1 192.168.255.1/24
route add default gw 192.168.255.2
route add -net 192.168.0.0/24 gw 192.168.1.254
echo 1 > /proc/sys/net/ipv4/ip_forward
exit 0
```

Serveur de Base de Données

Nous faisons en sorte à ce que le serveur de Base de Données ne puisse communiquer avec personne sauf le serveur Web, pour des raisons de sécurité.

Nous ajoutons aussi une exception pour 127.0.0.1 dans le cas où le ServeurBD devrait faire une requête à lui-même, ce qui ne crée pas de failles de sécurité.

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.
ifconfig eth0 192.168.1.20/24
iptables  -P INPUT DROP
iptables -I INPUT -s 192.168.1.10 -j ACCEPT
iptables -I INPUT -s 127.0.0.1 -j ACCEPT
exit 0
```

Serveur Web

Le serveur Web communiquera par défaut avec le Routeur2. Pour la sécurité du serveur Web, nous devrions bloquer l'accès aux ports non-utilisés, en gardant simplement les ports 20,21,22,80,443, mais nous rencontrons actuellement quelques problèmes car le serveur Web ne sera plus capable de faire des requêtes extérieures vers Iparla par exemple, car ces ports-là sont « aléatoires » et ne peuvent pas être débloqués en conséquence sans paramétrage avancé.

```
  GNU nano 2.2.6            File: /etc/rc.local

#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.
ifconfig eth0 192.168.1.10/24
route add default gw 192.168.1.1
exit 0
```

3ème jalon : Instalation des Services

Serveur de Base de Données

Dans un premier temps il va falloir modifier le l'interface d'écoute du service de base de données mySQL, Pour cela nous devons modifier dans le fichier ***/etc/mysql/my.cnf*** la ligne **:**

Bind-address = 0.0.0.0

Cette modification permet au service d'écouter sur toutes les interfaces réseau de la machine et plus uniquement 127.0.0.1

Nous pouvons maintenant passer à la configuration de la base de données en elle-même

Avec la commande **mysql -p** puis en entrant le mot de passe **toor**,

Nous nous retrouvons maintenant dans l'interpréteur de commande mySQL dans lequel nous allons exécuter les commandes suivantes :

- `CREATE DATABASE ndargazan001_bd;`
  Cette commande crée une nouvelle base de données

- `USE ndargazan001_bd;`
  Cette commande permet de sélectionner la base de données que nous souhaitons modifier

- `CREATE USER 'nicolas'@'192.168.1.10' IDENTIFIED BY 'motdepasse';`
  Cette commande créer un nouvel utilisateur nicolas avec un mot de passe et une autorisation de se connecter sur une machine d'adresse 192.168.1.10 (ServeurWEB)

- `GRANT ALL PRIVILEGES ON * . * TO 'nicolas'@'192.168.1.10';`
  Cette commande donne toutes les autorisations à l'utilisateur nicolas

Serveur Web

Dans un premier temps nous allons installer les différentes applications nécessaires

Avec **aptitude** et **apt** :

- `Aptitude install apache2`
- `apt-get install php5  php5-mysql`
  Ces commandes vont installer les modules Apache2 ainsi que PHP sur la machine
  D'autre librairies sont nécessaires mais sont installés automatiquement avec celles-ci

Nous pouvons maintenant démarrer le serveur web Apache2 avec la commande

/etc/init.d/apache2 start

Nous pouvons maintenant modifier le fichier **/var/www/index.html** en **/var/www/index.php** et ajouter les lignes suivantes :

```php
<?php
$database = new PDO('mysql:host=192.168.1.20;dbname=ndargazan001_bd;charset=utf8','nicolas',
'motdepasse');
$req=$database->prepare('SELECT * FROM `UTILISATEUR`');
$req->execute();
$resultat=$req->fetchAll();
foreach($resultat as $ligne)
{
        echo $ligne['nom'];
}

?>
```

Si nous partons du principe qu'une Table Utilisateur avec des occurrences existe dans la base de données nous les verrons s'afficher sur la page web avec la commande :

- `lynx 192.168.1.10`

Nous pouvons aussi activer le service SSH sur le serveur WEB ce qui nous permettra d'utiliser la fonction **sftp** pour y transférer des fichiers, simplement avec la commande **/etc/init.d/ssh start**

4ème jalon : Proof of concept

Dans ce jalon nous devons mettre en place un POC (Proof Of Concept) qui montre que nos différentes infrastructures fonctionnent et distribuent bien un service de serveur Web et de Base de données.

Serveur de Base de Données

Dernièrement nous avons créé une base de données avec un utilisateur autorisé d'interagir avec le serveur de BD pour tester que les communications fonctionnent.

Il nous suffit maintenant de créer des vraies tables que notre serveur web pourra interroger.

Nous avons élaboré un script SQL (Qui sera en annexe) permettant de créer les tables et de les peupler.

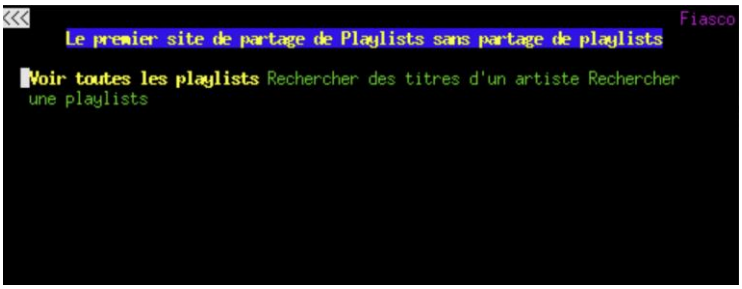
Pour exécuter le script dans l'interface mysql nous avons juste entré la commande :

- `source /root/script.sql`

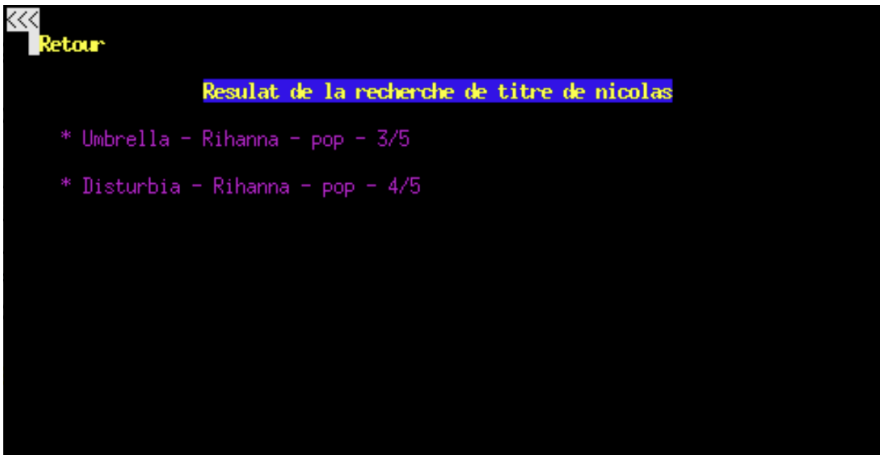
Serveur WEB

Les consignes indiquent que le serveur WEB doit servir une application de gestion de playlists permettant de récupérer les playlists d'un utilisateur (pour simplifier l'utilisateur ici est celui de la base de données). De plus cette application doit proposer un service de parcours de plusieurs pages PHP différentes, ici nous avons 3 possibilités de recherche :

- Afficher toutes les playlists avec leurs morceaux
- Sélectionner une playlist pour afficher ses morceaux
- Récupérer tous les morceaux d'un artiste



Page d'accueil

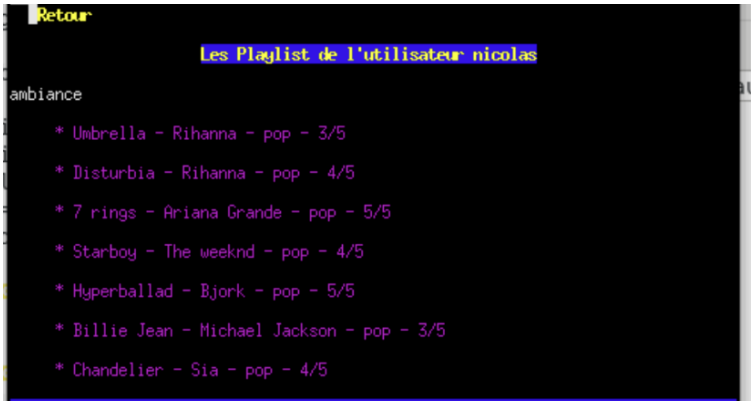


Recherche par artiste

Pour montrer les capacités de recherche en fonction d'une entrée utilisateur, nous avons mis en place une recherche de titre en fonction de l'artiste.

Ici initialiser une recherche avec l'artiste Rihanna nous retournera tous ses titres actuellement dans la base de données.

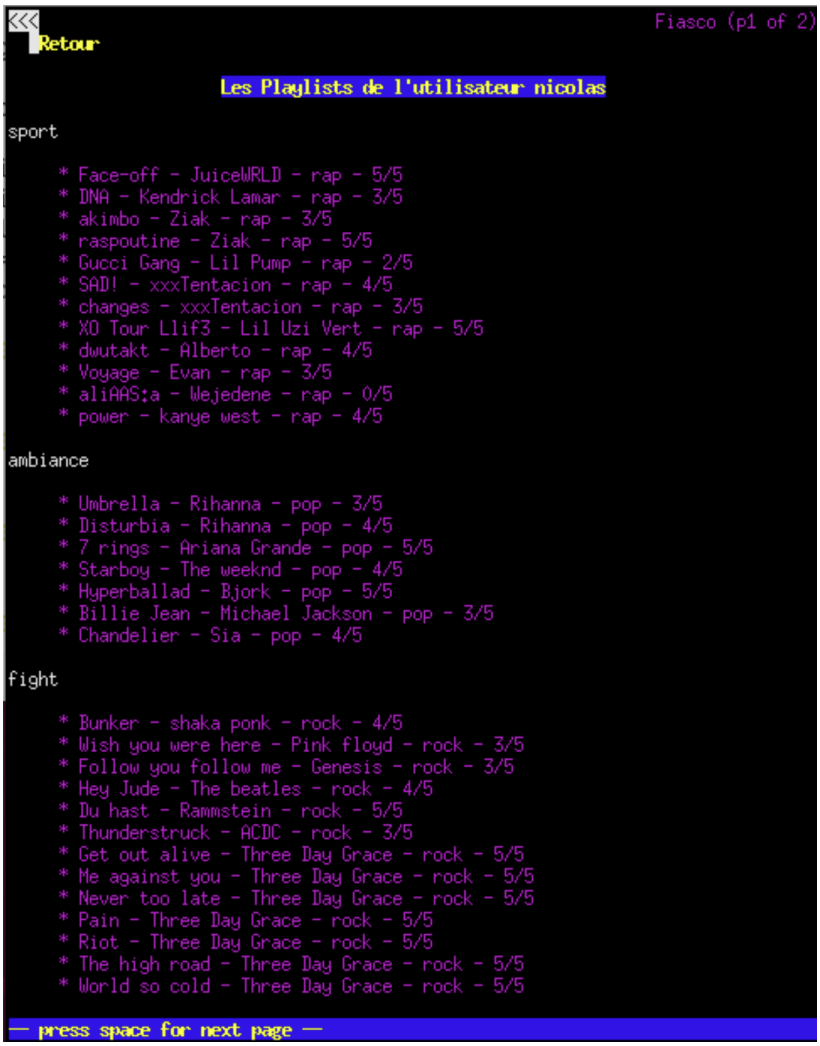
La recherche fonctionne même avec des parties de clé (comme « Rih » nous retourna tous les titres des artistes avec « rih » dans leur nom



Sélection d'une playlist parmi celles existantes

Pour utiliser de façon structuré la base de données on peut simplement effectuer une recherche par rapport à une playlists et récupérer ainsi tous les titres de celle-ci.

Dans cet exemple nous sélectionnons la playlists « ambiance » et nous nous retrouvons avec tous les titres de celle-ci



Affichage de toutes les playlists ainsi que leurs morceaux

La dernière fonctionnalité est juste une récupération totale de toutes les données musicales de la base de données.

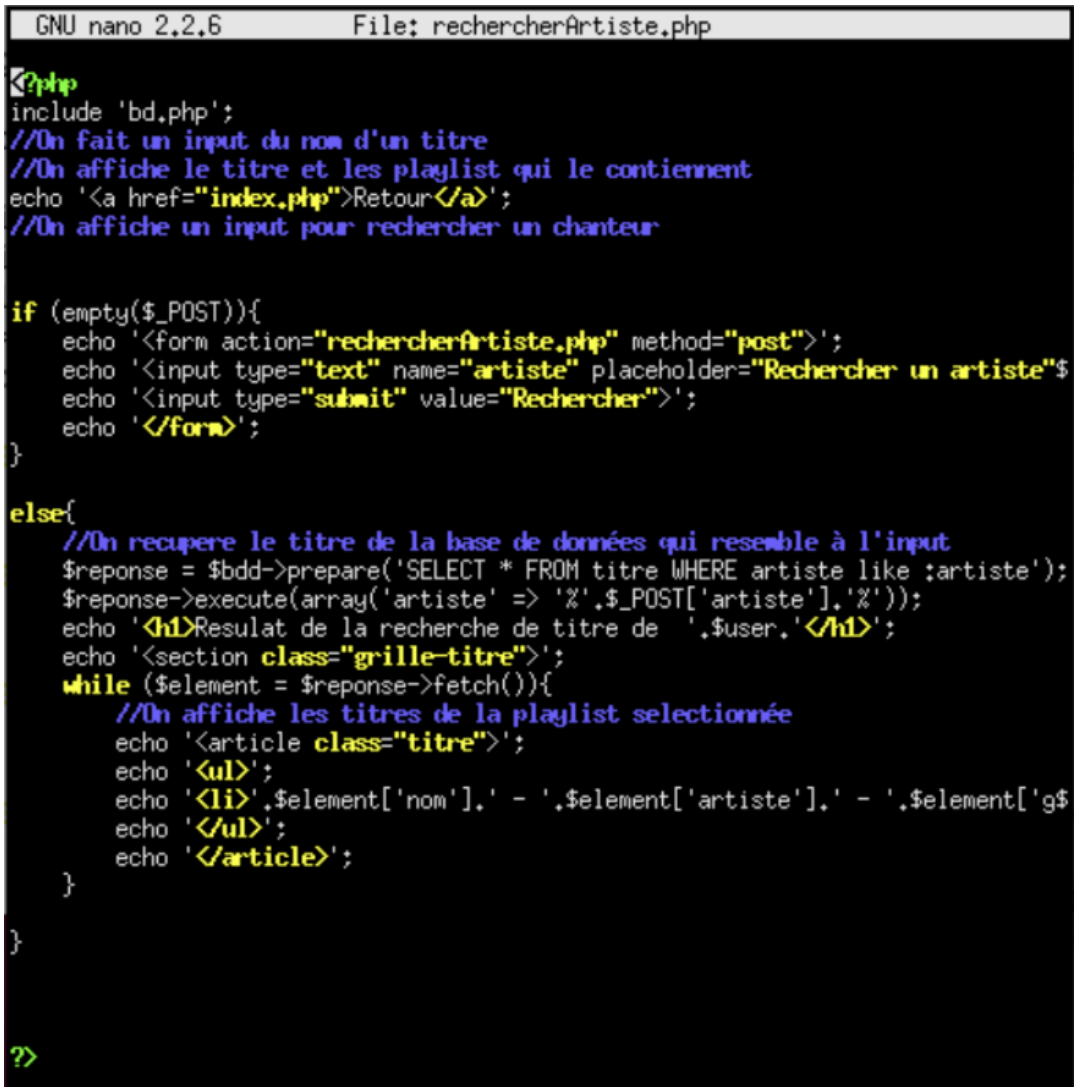
Simpliste celle-ci nous permet juste de vérifier que les données ont du sens et qu'il n'y avait pas de doublons involontaire (il est cependant possible de mettre le même titre dans deux playlists différentes)

Les fichiers PHP :

```
 GNU nano 2.2.6                File: index.php                          Modified

<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Fiasco</title>
</head>
<body>
    <h1>Le premier site de partage de Playlists sans partage de playlists</h1>
    <a href="playlist.php">Voir toutes les playlists</a>
    <a href="rechercherArtiste.php">Rechercher des titres d'un artiste</a>
    <a href="rechercherPlaylist.php">Rechercher une playlists</a>

</body>
</html>
```

La page d'accueil

Peu de complexité dans ce fichier, seulement une page « hub » permettant de rediriger sur les différentes fonctionnalités proposées par l'applications.

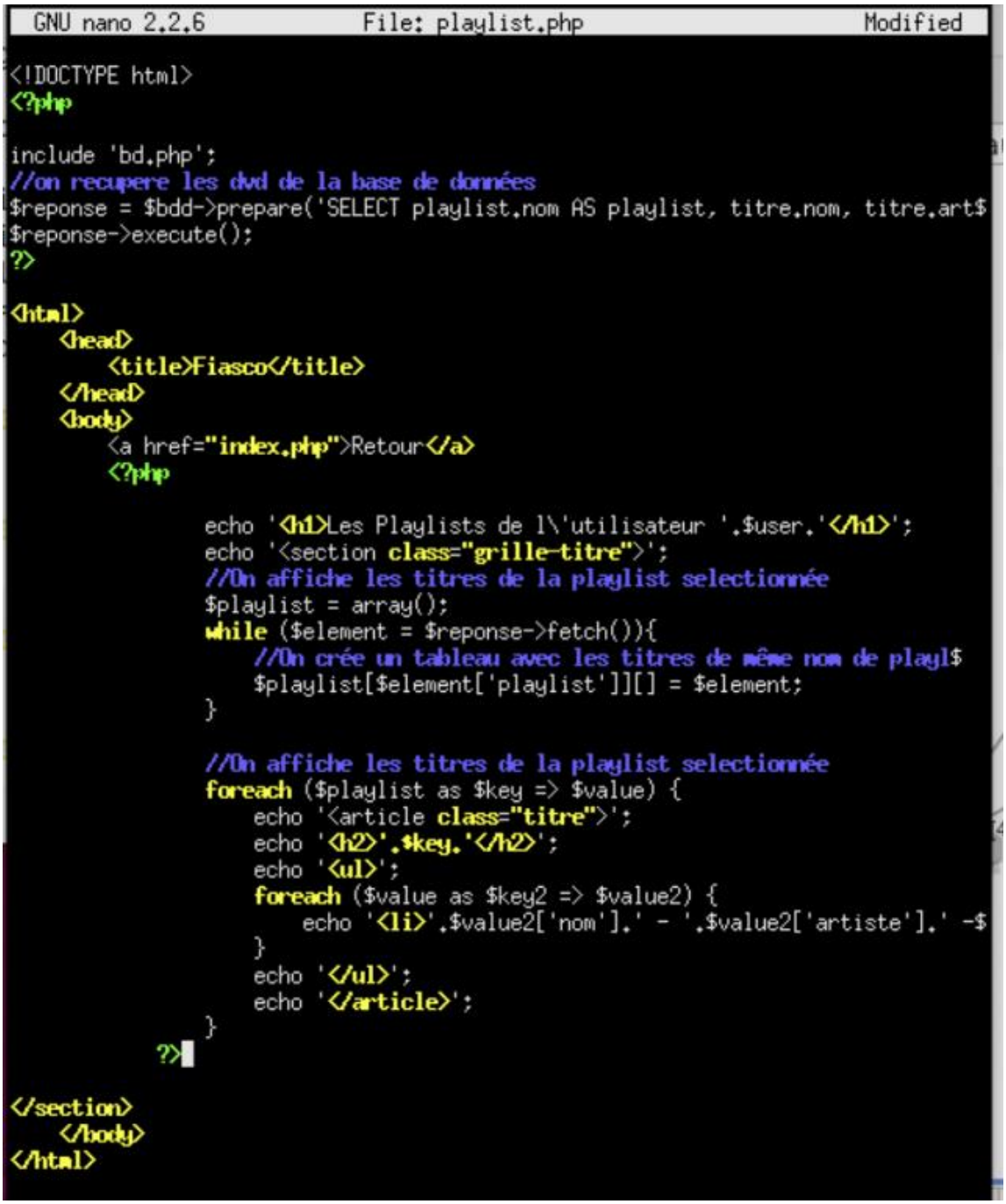


```
 GNU nano 2.2.6          File: rechercherArtiste.php

<?php
include 'bd.php';
//On fait un input du nom d'un titre
//On affiche le titre et les playlist qui le contiennent
echo '<a href="index.php">Retour</a>';
//On affiche un input pour rechercher un chanteur


if (empty($_POST)){
    echo '<form action="rechercherArtiste.php" method="post">';
    echo '<input type="text" name="artiste" placeholder="Rechercher un artiste"$
    echo '<input type="submit" value="Rechercher">';
    echo '</form>';
}

else{
    //On recupere le titre de la base de données qui resemble à l'input
    $reponse = $bdd->prepare('SELECT * FROM titre WHERE artiste like :artiste');
    $reponse->execute(array('artiste' => '%'.$_POST['artiste'].'%'));
    echo '<h1>Resulat de la recherche de titre de  '.$user.'</h1>';
    echo '<section class="grille-titre">';
    while ($element = $reponse->fetch()){
        //On affiche les titres de la playlist selectionnée
        echo '<article class="titre">';
        echo '<ul>';
        echo '<li>'.$element['nom'].' - '.$element['artiste'].' - '.$element['g$
        echo '</ul>';
        echo '</article>';
    }

}



?>
```

La page de recherche par artiste

Le fichier comporte deux affichages:

- Un formulaire si aucun donnée POST n'est reçu
- Une liste de titre résultant de la requête SQL



La page affichant toutes les playlists et leurs titres

On fait effectuer une recherche SQL des playlists ainsi que de leurs playlists. Pour ne pas surcharger la connexion, nous arrangeons toutes les données dans un tableau pour rassembler les titres avec leurs playlists.

On parcourt ensuite le tableau de manière à afficher la playlist et ses titres

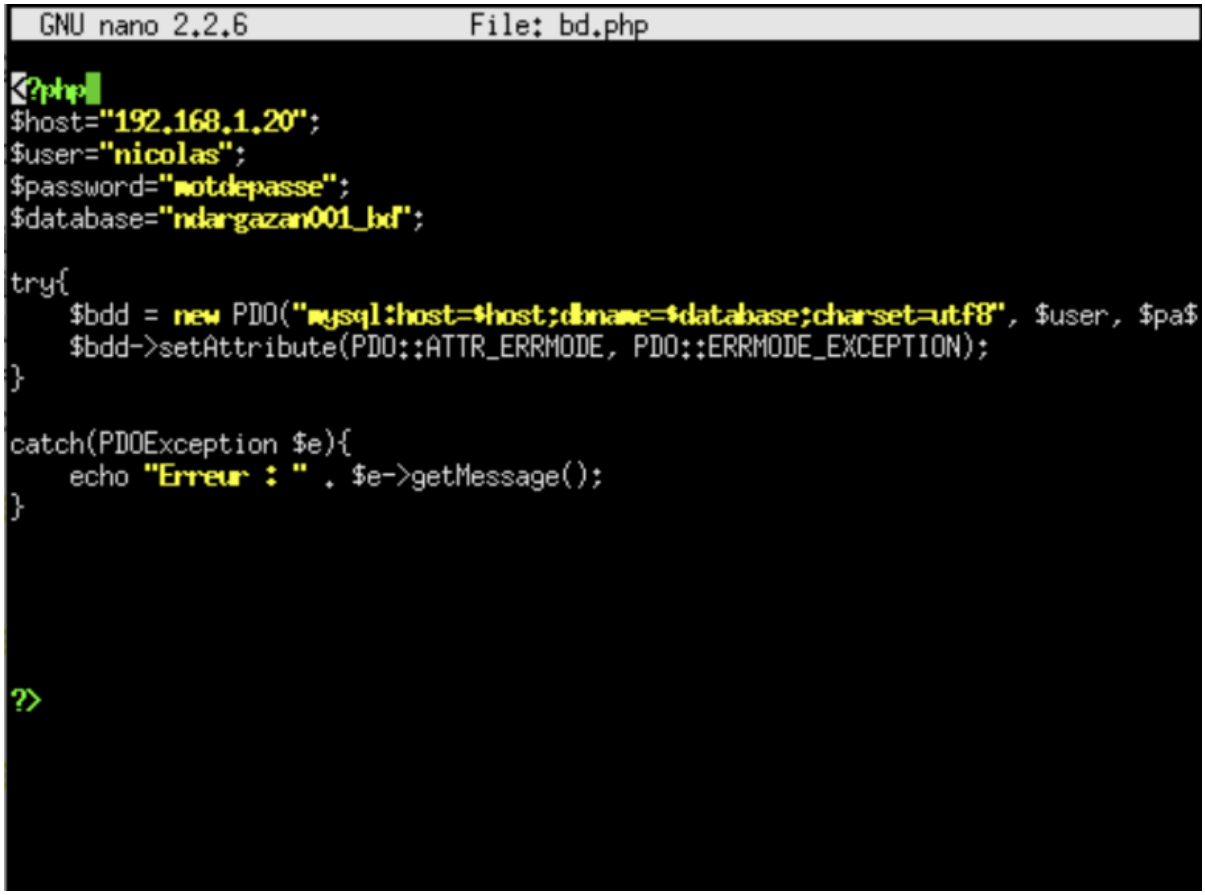
```
 GNU nano 2.2.6          File: rechercherPlaylist.php

<?php
include 'bd.php';
//On affiche 4 bouton des 4 playlist et il choisit
//On affiche les titres de la playlist selectionnée

//On recupere le nm des playlist de la base de données
$reponse = $bdd->prepare('SELECT nom FROM playlist');
$reponse->execute();

echo '<a href="index.php">Retour</a>';

if (empty($_POST)){
    //On affiche les playlist
    echo '<h1>Les Playlists de l\'utilisateur '.$user.'</h1>';
    echo '<section class="grille-titre">';
    while ($element = $reponse->fetch()){
        //On les affiches dans des boutons
        echo '<article class="titre">';
        echo '<form action="rechercherPlaylist.php" method="post">';
        echo '<input type="submit" name="playlist" value="'.$element['nom'].'">$
        echo '</form>';
        echo '</article>';
    }
}

else{
    $reponse = $bdd->prepare('SELECT playlist.nom AS playlist, titre.nom, titre$
    $reponse->execute(array('playlist' => $_POST['playlist']));
    echo '<h1>Les Playlist de l\'utilisateur '.$user.'</h1>';
    echo '<section class="grille-titre">';
    //On affiche les titres de la playlist selectionnée
    echo '<h2>'.$_POST['playlist'].'</h2>';
    while ($element = $reponse->fetch()){
        //On affiche les titres de la playlist selectionnée
        echo '<article class="titre">';
        echo '<ul>';
        echo '<li>'.$element['nom'].' - '.$element['artiste'].' - '.$element['g$
        echo '</ul>';
        echo '</article>';

    }
}

?>
```

La page de recherche par playlists

Le fichier comporte aussi deux affichages:

- Qui génère autant de formulaires que de playlists
  On aurait aussi pu (et surement dût) plutôt utiliser des requêtes GET avec des balises <a> au lieu de crée plusieurs formulaires sur une page.

- La liste de titres inclus dans la playlist envoyé dans le POST



Le fichier de connexion à la BD

Ce fichier n'est techniquement pas une page WEB mais il reste essentiel pour l'entièreté du fonctionnement du site, c'est celui-ci qui nous permet d'interagir avec la BD sur le serveur de base de données ;

Nous retrouvons les informations précédentes, comme l'adresse du serveur BD, ainsi que le nom et mot de passe de l'utilisateur auquel on a précédemment autorisé la connexion sur l'adresse du server web ? ainsi que le nom de notre base de données.

Nous utilisons PDO comme interface plutôt que mysqli simplement car nous avons rencontré beaucoup d'erreurs que nous n'avions pas avec PDO qui dispose aussi de commande plus facile à utiliser comme prepare() et execute()