Développement d'applications

relationnelles

Bureau des artistes

Giovanna Bonicelli

IG3

SOMMAIRE

1. Objectif.......... .......... .......... .......... .......... .......... .......... .......... .......... .......... 3

a. Cahier des charges.......... .......... .......... .......... .......... .......... .......... 3

b. Description projet.......... .......... .......... .......... .......... .......... .......... 3

1. La base de données.......... .......... .......... .......... .......... .......... .......... .......... 4

a. Présentation de la base de données.......... .......... .......... .......... .......... 4

b. Choix du SGBD.......... .......... .......... .......... .......... .......... .......... 6

1. Site web.................... .......... .......... .......... .......... .......... .......... .......... .......... 7

a. Choix du langage.......... .......... .......... .......... .......... .......... .......... 7

b. Explication du code.......... .......... .......... .......... .......... .......... .......... 8

1. Hébergeur.......... .......... .......... .......... .......... .......... .......... .......... .......... 9

a. Choix de l'hébergeur.......... .......... .......... .......... .......... .......... .......... 9

b. Schéma architecture technique.......... .......... .......... .......... .......... 10

1. Discussion et auto-critique.......... .......... .......... .......... .......... .......... .......... 11

Annexe.......... .......... .......... .......... .......... .......... .......... .......... .......... .......... 12

Liste des figures

Figure 1. MCD 5

Figure 2. Schéma d’architecture 10

Liste des tableaux

Tab.1 - Comparaison de plusieurs SGBD 6

Tab.2 - Langages envisagés pour le développement du site web 7

Tab.3 - Comparaison des hébergeurs 9

Ce document a pour objet de décrire le projet que j’ai réalisé dans le cadre des cours SQL et WEB de 3ième année de l'option Informatique et Gestion de Polytech Montpellier.

La première partie présente les objectifs du projet. Les deuxième et troisième parties décrivent les méthodes utilisées et discutent les outils choisis pour les développements SQL et HTML, respectivement. La quatrième partie présente le choix de l’hébergeur ainsi que le schéma technique de l’architecture. Enfin, un retour auto-critique fait l'objet de la cinquième partie.

1. Objectif

a. Cahier des charges

Le cahier des charges qui nous a été proposé comprenait deux parties :

* réaliser une base de données avec des contraintes statiques (clés primaires et étrangères) et dynamiques (deux triggers) contenant au moins trois tables et plus cinq cents n-uplets dans une des tables.
* développer un site web permettant de créer, de supprimer, de modifier et d'afficher un ou plusieurs n-uplets de notre base de données avec des URI propres avec la contrainte de ne pas utiliser de session pour ce site.

b. Description du projet

Au sein de Polytech Montpellier, le bureau des étudiants rassemble différents clubs, sportifs et artistiques, qui proposent différentes activités. L'application que j'ai créée porte sur le bureau des artistes qui s'occupe des clubs artistiques.

Les étudiants peuvent s'inscrire dans 7 clubs artistiques (Dramateck, Musiteck, Polybeats, Polycook, Polypix, Polypop et Politimes). Ces clubs proposent des évènements tout le long de l'année (la semaine des clubs, le Téléthon, le spectacle de noël, le Defistop, le festival de musique et le spectacle final) et les adhérents peuvent s'inscrire pour coopérer à leur organisation.

Mon application permet aux étudiants de s'inscrire à ces clubs et, une fois inscrits dans un club, de s'inscrire pour participer à l'organisation d'évènements. Elle permet d'afficher les listes des adhérents aux clubs artistiques, ainsi que les listes des étudiants participant à l'organisation des évènements. A l'inverse, il est possible d'afficher à quel(s) club(s) et quel(s) évènement(s) un étudiant donné est inscrit. L'application permet enfin de supprimer un étudiant d'un club ou d'un évènement ou de modifier l'identité d'un étudiant (si l'étudiant s'est trompé au moment de l'inscription).

**2. La base de données**

a. Présentation de la base de données

Pour modéliser ma base de données, j’avais besoin d’une méthode de modélisation. Cette année, nous avons appris deux méthodes : UML et Merise2. J’ai sélectionné Merise2 parce que j’étais plus à l’aise avec cette méthode.

Pour réaliser mon projet, je devais construire une base de données. Cette base de données relie toutes mes données selon un schéma qui permet de relier les caractéristiques d'un étudiant à celles d'un ou plusieurs clubs et celles d'un ou plusieurs évènements (cf. Fig.1).

Ma base de données comporte ainsi 5 tables :

-etudiant

-club

-evenement

-participe

-coopere

Les règles de gestion sont :

* Un étudiant peut participer à plusieurs clubs
* Un clubs comporte plusieurs étudiants
* Un étudiant peut coopérer plusieurs évènements s'il fait partie d'un club
* Un évènement est coopéré par plusieurs étudiants

Le schéma relationnel est :

etudiant(ine, option, annee, nom, prenom)

club(nom\_club)

evenement(numero\_ev, nom\_ev, heure, date, lieu)

participe(#ine, #nom\_club)

coopere(#ine, #numero\_ev)

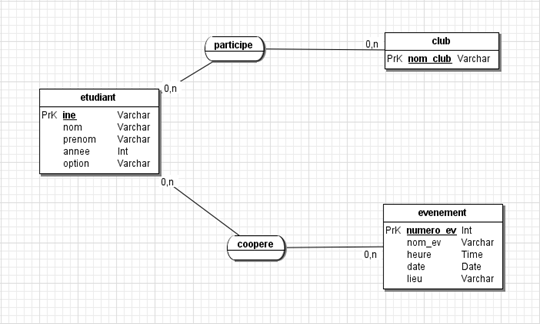
L'attribut « ine » correspond à une chaîne de caractères (comportant aussi des chiffres) ne pouvant dépasser 11 caractères.

L'attribut « annee » peut prendre les valeurs 3, 4 ou 5 (correspond aux années de cycle ingénieur).

L'attribut « option » peut prendre les valeurs ENR, IG, MAT, MI, MEA, STE, STIA, MSI, EGC ou SE.

L'attribut « numero\_ev » peut prendre les valeurs de 1 à 6 (il n'y a que 6 évènements).

Les noms des tables et les noms des attributs sont en minuscules et sans accent pour faciliter l'utilisation.



*Figure 1 . MCD*

La base de données contient deux triggers :

* le premier empêche de supprimer un étudiant d'un club s'il participe à un évènement (il utilise la procédure est\_dans\_club())
* le deuxième empêche d'insérer un étudiant dans un évènement s'il n'est pas inscrit dans un club (il utilise la procédure etud\_coopere()).

b. Choix du SGBD

Il me fallait un SGBD gratuit, dans lequil il soit possible de paramétrer des triggers et d’imposer des contraintes. Comme je souhaitais développer mon application sur mon PC personnel, il était nécessaire que le SGBD soit compatible avec le système d'exploitation Windows 8, qu’il soit gratuit et qui ne soit pas trop “gourmand” en ressources. Le tableau ci-dessous compare les différents outils que j'ai envisagé d'utiliser.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | MySQL | oracle | Informix | PostgreSQL |
| Demandeur de ressources | Connexion :  ~ 100 ko /utilisateur | Connexion : 700ko/utilisateur (mémoire) et espace disque | ne demande pas beaucoup de resssources | ne demande pas beaucoup de ressources |
| Facilité d'utilisation | +  interface graphique | +  appris en cours | +  appris en cours | +  conseils disponibles par relation |
| hébergement | oui | Service web | non | Pas de service web |
| extension | Bonne interaction apache/php | PL-SQL | PL-SQL, datablade text,  datablade spatiale | Java, Ruby, PL-SQL |
| Performance |  | + | + | + |
| prix | gratuit | gratuit | il existe une edition gratuite | gratuit |
| Trigger | Support incomplet | oui | oui | oui |
| Disponibilité | Linux, Windows, MacOSX, Unix, BSD, OS2 | Linux, Windows, Unix, MacOSX | Unix, Windows, BSD, MacOSX | Linux, Unix, MacOSX, Windows |

Tab.1 – Comparaison de plusieurs SGBD. Le signe ‘+’ est un indicateur de performance et de facilité d’utilisation.

Dans ce tableau, ne sont pris en compte que les éléments en rapport avec mon projet. Parmi les 4 outils envisagés, j’ai retenu par élimination PostgreSQL qui correspondait bien à mon application. J’ai rejeté Oracle parce qu’il demande des ressources trop importantes, Informix par que l’outil graphique pour l’administration est payant et MySQL à cause du support pour les triggers qui est incomplet.

Les avantages et inconvénients de chaque outil sont détaillés dans l’annexe (partie 1).

3. Site WEB

Un site web est un ensemble de page web. Une page web est constituée de documents codés en HTML(base du site), CSS(mise en forme du site) et d'images. Le langage HTML permet aux navigateurs Web d'adapter le rendu de chaque page au navigateur web utilisé (Chrome, Firefox, etc.). Le cahier des charges impose une connexion à une base de donnée. Pour cela il faut créer une page web dynamique (c'est une page qui varie en fonction des informations qui ne sont connues qu'au moment de sa consultation). Pour générer ces pages web, il faut utiliser un langage spécifique (PHP, Ruby, Python ..) car HTML et CSS ne suffisent pas.

a. Choix du langage

Plusieurs langages sont disponibles pour développer des pages web. En un premier temps j’ai écarté Java car, bien que réputé pour sa stabilité, il ne me permettait pas d’interagir avec ma base de données. Il aurait par contre permis de vérifier les éléments d'un formulaire sans rafraîchir la page, ajouter des animations, etc. Les langages retenus permettent d'interagir avec une base de données, donc stocker des informations comme des news, des messages sur les forums, des membres etc ...

J’ai comparé dans le tableau 2, trois langages : PHP, Python et Ruby. On peut remarquer que les 3 sont des langages de script ce qui amoindrit leurs performances mais facilite leur utilisation.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Php | Ruby | Python |
| Principale caractéristique |  | programmation agréable et flexible | code très lisible |
| Facilité d'utilisation | + | ++ | + |
| Facilité d’apprentissage | ++  Très bien documenté | + si on connaît un ou deux langages | +++ |
| Nombre de lignes de codes | + | ++ | + |
| Temps d’exécution moyen | Plutôt lent | Intermédiaire | Plutôt rapide |

Tableau 2 – Langages envisagés pour le développement du site web, + représente le niveau de facilité (+++: très facile, +:facile) et le nombre de lignes de code (+++: beaucoup, +:peu)

Je connaissais déjà python et je voulais apprendre un nouveau langage : PHP m'a paru être plus adapté que Ruby parce qu’il est plus facile d’apprentissage et mieux documenté.

Les sources utilisées pour cette comparaison sont indiquées dans l’annexe, partie 2.

Pour la mise en forme de mon site web plusieurs langages pouvaientt être utilisé. J’avais sélectionné Bootstrap et AngularJS. Je n’ai ulisé que Bootstrap car AngularJS ne m’était pas utile pour répondre au cahier des charges. Cependant si j’avais eu plus de temps je l’aurai utilisé. AngularJS permet d’avoir une application maintenable, modulable et organiser (car une application ne comporte pas qu’une seule page web et requiert beaucoup de javascript, si on utilise ce langage).

b. Explication du code

Pour me connecter à ma base de données avec PHP j'ai utilisé l'extension PDO. C'est un outil qui permet d'accéder à n'importe quel type de base de données (MySQL, PostgreSQL, Oracle). L’ensemble des fichiers qui codent le site est fourni dans l'annexe, partie 5.

Mon application permet de modifier, de supprimer, de rechercher et de créer des étudiants. Le détail de ces fonctionnalités est expliqué ci-dessous.

Rechercher

- Rechercher les clubs d'un étudiant (<https://bda-bonicelli.herokuapp.com/club.php>)

Il est proposé aux utilisateurs de consulter la liste des participants de chaque club (trié par club) en cliquant sur 'clubs' dans le menu puis sur 'Les clubs'.

Pour pouvoir afficher cette liste, il est nécessaire d'afficher la table 'participe'.

Pour pouvoir afficher les clubs d'un étudiant, j'ai utilisé un formulaire qui permet à l'utilisateur d'entrer le nom et le prénom de la personne qu'il cherche. Les résultats du formulaire sont récupérés sur un fichier cible ('recherche.php') qui exécute les requêtes SQL (grâce à la fonction 'query').

-Rechercher les évènements auxquels coopère l'étudiant ([https://bda-bonicelli.herokuapp.com/evenement.php](https://bda-bonicelli.herokuapp.com/club.php))

Il est proposé aux utilisateur de consulter la liste des participants de chaque club (trié par club) en cliquant sur 'évènement' dans le menu puis sur 'Les évènements'.

Pour pouvoir afficher cette liste, il est nécessaire d'afficher la table 'coopere'.

Pour pouvoir afficher les évènements auxquels coopère un étudiant, j'ai utilisé un formulaire qui permet à l'utilisateur d'entrer le nom et le prénom de la personne qu'il cherche. Les résultats du formulaire sont récupérés sur un fichier cible ('rechercheev.php') qui exécute les requêtes SQL

Créer

- Pour qu'un utilisateur s’inscrive à un ou plusieurs clubs il doit cliquer sur 'inscription' dans le menu puis sur 'club' (<https://bda-bonicelli.herokuapp.com/inscription.php>).

Il lui est demandé son numéro INE, son nom, son prénom, son option, son année et les clubs auxquels il veut participer.

Si l’une des informations n’a pas été saisie, le formulaire est renvoyé à l’utilisateur avec un message d’erreur indiquant les informations qu’il n’a pas fournies.

Les clubs sont représentés par des “checkbox”. Si un club n'est pas sélectionné alors la requête SQL le concernant ne sera pas exécutée.

Le principe est le même que pour 'rechercher', avec un formulaire et un fichier cible ('cible.php').

Une fois le fichier cible exécuté le nouvel étudiant a été rajouté à la table 'etudiant' et à la table 'participe' pour les clubs qu'il a sélectionnés.

- Pour qu'un utilisateur coopère à un ou plusieurs évènements il doit cliquer sur 'inscription' dans le menu puis sur 'évènement' (<https://bda-bonicelli.herokuapp.com/inscriptionev.php>).

Il lui est demandé son numéro INE et les évènements pour lesquels il veut coopérer.

S'inscrire à un évènement fonctionne de la même manière que pour un club.

Le fichier cible est 'cibleev.php'.

Une fois le fichier cible exécuté le nouvel étudiant a été rajouté à la table 'coopere' pour les évènements qu'il a sélectionnés.

Cependant lors d'une insertion à 'coopere', un trigger vérifie que l'étudiant participe à un club : s'il ne participe à aucun club alors l'étudiant ne sera pas rajouté à la table et le formulaire sera renvoyé à l'utilisateur avec le message d'erreur (qui lui expliquera pourquoi cela n'a pas fonctionné).

Supprimer

-Pour supprimer sa participation à un club, l'étudiant devra cliquer sur 'club' dans le menu puis sur 'supprimer votre participation à un club' (<https://bda-bonicelli.herokuapp.com/supprimerclub.php>)

Il lui sera demandé son nom, son prénom, son numéro INE et les clubs auxquels il ne veut plus participer (sous forme de chexkbox).

Le principe est le même que pour 'créer' à l'exception que l'étudiant n'est pas inséré dans la table 'partipe' mais supprimé.

Un trigger contrôle si l'étudiant coopère à un évènement. Le cas échéant, il ne sera pas possible de le supprimer du club (et le formulaire sera renvoyé à l'utilisateur avec le message d'erreur ).

-Pour supprimer sa coopération à un évènement, l'étudiant devra cliquer sur 'évènement' dans le menu puis sur 'supprimer votre participation à un évènement' ([https://bda-bonicelli.herokuapp.com/supprimerev.php](https://bda-bonicelli.herokuapp.com/supprimerclub.php)).

Il lui sera demandé son nom, son prénom, son numéro INE et les évènements auxquels il ne veut plus coopérer (sous forme de checkbox).

Le principe est le même que 'créer' à l'exception que l'étudiant n'est pas insérer dans la table 'coopere' mais supprimer.

Modifier

Pour modifier son identité (ine) un utilisateur doit cliquer sur 'inscription' dans le menu puis sur 'modifier identité' (<https://bda-bonicelli.herokuapp.com/modification.php>).

Il doit remplir un formulaire avec son nom, son prénom et son numéro INE. Un nouveau formulaire est alors envoyé avec son nom, son prénom, son numéro INE, son option et son année. Son ancienne identité est supprimée des tables 'etudiant', 'coopere' et 'participe'. Sa nouvelle identité est ensuite insérée dans 'etudiant' et les anciennes liaisons des clubs et des évènements à son ancien numéro INE sont insérées mais avec son nouveau numéro INE.

**4. Hébergeur**

Un hébergeur permet de mettre à disposition des internautes des sites web. J'ai donc choisi un hébergeur et y ai déposé mon site.

a. Choix de l’ hébergeur

J'ai sélectionné trois hébergeurs gratuits différents : Hostinger, Olympe, Heroku. Dans le tableau ci-dessous j'ai comparé ces trois hébergeurs par rapport aux éléments que je considère les plus importants pour mon projet.

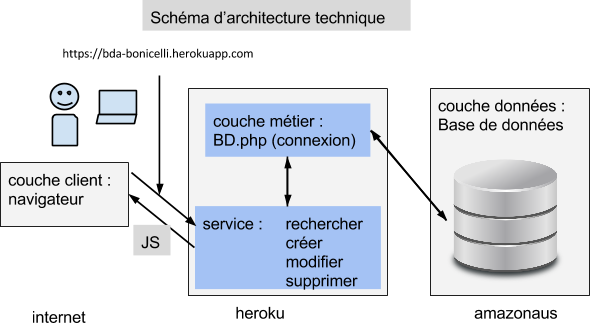
|  |  |  |  |
| --- | --- | --- | --- |
|  | Hostinger | Olympe | Heroku |
| SGBD | MySQL | MySQL, PostgreSQL | MySQL, Postgres |
| Installation | facile | facile | facile |
| trigger | oui | non | oui |

Tableau 3 – Comparaison des hébergeurs

J' ai utilisé PostgreSQL pour ma base de données donc je ne pouvais pas choisir Hostinger. Les triggers ne sont pas pris en compte par Olympe et j'ai donc choisi Heroku. De plus Heroku utilise un dépôt Git, ce qui m’a permis de m’entraîner à l’utilisation de Git.

Les sources du tableau figurent dans l’annexe, partie 3.

b. Schéma architecture technique



*Figure 2. Schéma d’architecture*

Cette figure représente les différentes couches de mon application. La couche client correspond à la présentation des données. La couche métier représente le traitement métier des données. La couche concerne l’accès au données. Les couches communiquent entre elles et chacune d'entre elles propose un ensemble de services rendus. Les services d'une couche sont mis à disposition de la couche de gauche.

**5. Discussion et auto-critique**

Le projet que j’ai développé répond au cahier des charges excepté pour les URI propres car j'ai découvert trop tard qu'il fallait le faire.

M'auto-former m'a pris beaucoup de temps mais je pense que je n'aurais pas appris aussi vite en cours. Ce qui prend le plus de temps n’est pas l’apprentissage en tant que tel mais la correction des problèmes (‘bugs’) qui demande de l’expérience. Quand on n’a pas encore cette expérience, il est précieux de pouvoir se renseigner auprès de personnes qui ont déjà rencontré les mêmes erreurs, soit dans les forums sur internet (mais l’information n’est pas toujours accessible, ou facile à trouver), soit auprès de personnes qui ont déjà résolu ces problèmes. Ainsi, je pense qu’il est important d’avoir manipulé un maximum d’outils et de garder la trace des problèmes rencontrés car cela permettra de gagner beaucoup de temps quand il faudra les utiliser dans un contexte professionnel.

Avec plus de temps, j'aurais souhaité :

* rajouter une table dans ma base de données qui aurait relié les clubs aux évènements (seuls certains clubs auraient pu organiser des évènements)
* faire un MVC
* faire un formulaire pour une connexion pour les utilisateurs sur l'application : un utilisateur serait le seul responsable de son compte (seul cet utilisateur pourrait modifier son identité, créer son compte, se supprimer des évènements et des clubs) grâce à un mot de passe.

Les choix d’outils que j’ai réalisés se sont révélés être bien adaptés pour ce projet. J’aurais aimé utiliser AngularJs mais le temps donné pour faire ce projet ne me le permettait pas.

**ANNEXE**

1. Comparaison des SGBD

MySQL

**Disponibilité** : Linux, Windows, MacOSX, Unix, BSD, OS2

**Avantages**

* Solution très courante en hébergement public
* Très bonne intégration dans l'environnement Apache/PHP
* Facilité de déploiement et de prise en main.

**Inconvénients**

* Ne supporte qu'une faible partie des standards SQL-92
* Support incomplet des triggers et procédures stockées
* Pas d'héritage de table
* Pas de vue matérialisée

ORACLE

**Disponibilité :**Linux, Windows, Unix, MacOSX

Oracle n'est pas optimisé pour de petites bases de données. Son intérêt est démontré pour les grands volumes de donnée (>200Go) et pour un grand nombre d'utilisateurs (>300).

**Avantages**

* Richesse fonctionnelle
* Procédures stockées en PL-Sql (langage propriétaire Oracle, orienté ADA) ou en JAVA ce qui peut s'avérer utile pour les équipes de développement.
* Pérennité de l'éditeur : avec plus de 40% de part de marché.
* Accès aux données système via des vues, bien plus aisément manipulable que des procédures stockées.
* Services Web, support XML

**Inconvénients**

* Fort demandeur de ressources, Oracle est bien plus gourmand en ressource mémoire que ses concurrents, ce qui implique un investissement matériel non négligeable. La connexion utilisateur nécessite par exemple près de 700 Ko/utilisateur, contre une petite centaine sur des serveurs MS-SQL.
* Gourmand aussi en espace disque

Informix

Informix

**Disponibilité :**Unix, Windows, BSD, MacOSX

**Avantages**

* Administration simpliste et efficace
* Performant
* Stabilité

**Inconvénients**

* Pérennité de la solution : on ne sait pas si cet outil pas perdurer.
* Très peu d'outils graphiques disponibles pour l'administration (hormis Server Studio JE payant)

Postgre-SQL

**Disponibilité :** Linux, Unix, MacOSX, Windows

**Avantages**

* Fiable et relativement performant, tout en restant simple d'utilisation
* Supporte la majorité du standard SQL-92 et possède en plus un certain nombre d'extensions (Java, Ruby, PL-SQL).
* Très riche fonctionnellement, notions d'héritage de tables, multitude de modules
* Simple d'utilisation et d'administration
* Héritage de tables

I**nconvénients**

* Supporte les bases de moyenne importance
* Pas de services Web
* Pas de vue matérialisée

Sources : <http://fadace.developpez.com/sgbdcmp/#LII-F-4>

2. Sources pour la comparaison des langages web

[http://www.hosting-review.com/euro-fr.php?aff\_sub=H-R-Europe-FR&aff\_sub2=meilleurs%20h%C3%A9bergeurs%20web&aff\_sub3=b&gclid](http://www.hosting-review.com/euro-fr.php?aff_sub=H-R-Europe-FR&aff_sub2=meilleurs%20hébergeurs%20web&aff_sub3=b&gclid)={gclid}&ga1=H-R-Europe-FR&ga2=meilleurs%20h%C3%A9bergeure%20web&ga3=b&ga4=adwords&gclid=CNipqOHh68UCFQXLtAod4zYAwQ

<http://willdurand.fr/programmation-web-comment-choisir-entre-le-confort-et-les-performances/>

<http://www.rankspirit.com/langage.php>

3. Sources pour la comparaison des hébergeurs

<http://www.commentcamarche.net/faq/1624-choisir-un-hebergeur-gratuit-ou-payant>

<http://www.quora.com/What-are-the-advantages-and-disadvantages-of-using-heroku>

<http://openclassrooms.com/forum/sujet/heberger-votre-site-web-gratuitement-64210>

4. Connexion à un base de données avec php

<?php

// Sous WAMP (Windows)

$bdd = new PDO('mysql:host=localhost;dbname=test;charset=utf8', 'root', '');

// Sous MAMP (Mac)

$bdd = new PDO('mysql:host=localhost;dbname=test;charset=utf8', 'root', 'root');

?>

5. Présentation des fichiers du site web

Le code du site web est constitué de plusieurs fichiers .php :

* **entete.php** contient le type de caractère, les liens vers mes fichier css et mon menu (**menu.php**)
* **pied\_de\_page.php** contient les liens vers les fichiers javascript (le jquery qui permet d’exécuter les requêtes sql et le bootstrap qui permet d'utiliser bootstrap)
* ces deux fichiers php sont intégrés dans tous les fichiers php qui correspondent à une page de mon application
* **BD.php** permet de se connecter à ma base de données. Tous les fichiers qui contiennent des requêtes sql contiennent BD.php
* **index.php** correspond à ma page d’accueil
* **club.php** contient **rechercher.php** (qui permet de rechercher un étudiant avec les clubs dans lesquels il est) et tous les étudiants qui participent à des clubs (triés par club). Il correspond au bouton 'les clubs' dans le dropdown 'clubs' dans le menu.
* idem pour **evenement.php** sauf qu'il correspond aux évènements et non au club
* **supprimerclub.php** permet de supprimer un étudiant d'un club (la suppression se fait grâce à **cible\_supprimer\_club.php**)
* idem pour **supprimerev.php** et **cible\_supprimer\_ev.php**
* **inscription.php** ( bouton 'club' dans le dropdown 'inscription' dans le menu) permet de rajouter dans les tables etudiant et participe l'étudiant (grâce à **cible.php**)
* **inscriptionev.php** ( bouton 'évènement' dans le dropdown 'inscription' dans le menu) permet de rajouter dans la table coopere l'étudiant (grâce à **cible\_ev.php**)
* **modification.php** (bouton 'modifier identité' dans le dropdown 'inscription' dans le menu) permet de modifier l'identité d'un étudiant dans la table etudiant