

Affinity Finder

Projet informatique Base De Données Web

David Leroy
ENSAE ParisTech

Louis Tassin de Charsonville
ENSAE ParisTech

Résumé

La présente note détaille notre projet informatique de base de données Web en présentant les objectifs, la démarche et explique la structure du code. Un schéma représentant l'architecture du site et des prises écran sont situés en annexe.

Nous présentons le fonctionnement de notre site, Affinity Finder, qui permet à ses utilisateurs de trouver le meilleur partenaire pour un projet d'étude.

Les langages utilisés dans le cadre de ce projet sont les suivants : principalement HMTL, PHP, MySQL et de manière subsidiaire : Python, Javascript, CSS.

1 Objectifs du projet

L'objectif de notre projet informatique était de créer un site internet permettant aux élèves de trouver un partenaire pour les différents projets réalisés dans le cadre de l'école. Trouver un bon partenaire dépend de nombreuses caractéristiques et notre site aide les élèves à trouver un partenaire parmi les élèves qui leur ressemblent.

Concernant l'aspect "programmation", l'objectif était de faire un site dynamique interagissant avec une base de données relationnelle comprenant plusieurs tables. Construit selon le schéma d'un réseau social, notre site s'appuie sur une structure simple : les utilisateurs se connectent à leur profil via une page de connexion et peuvent parcourir la table des autres utilisateurs à travers une fonction de recherche qui mesure la distance euclidienne entre les utilisateurs grâce à leurs coordonnées dans l'espace des préférences. La communication entre les élèves est assurée grâce à un service de messagerie.

2 Architecture et structure du site

Le site est composé d'une dizaine de pages et d'une base de données comprenant deux tables (un schéma en annexe résume l'architecture du site). Toutes les pages sont codées en PHP et l'interaction avec la base de données se fait via des requêtes SQL. Les feuilles de style (.css) utilisées sont celles du package Bootstrap (<http://twitter.github.io/bootstrap/>).

On peut regrouper les pages de notre site en quatre pages principales :

- une page de connexion
- une page de profil
- une page "messages"
- une page recherche

Et une base de données comprenant deux tables :

- la table contenant les profils et caractéristiques de chaque utilisateur. La clé primaire est le pseudo de l'utilisateur.
- la table contenant les messages que s'envoient les utilisateurs. La clé primaire est l'identifiant associé au message (un entier).

2.1 La base de données

2.1.1 La table 'users'

La table 'users' regroupe l'ensemble des utilisateurs du site Affinity Finder. À chaque utilisateur correspond une ligne avec les différentes caractéristiques des utilisateurs (cf Table 1) ainsi que le chemin vers leur photo de profil.

La table 'users' n'existe pas lors de la première connexion au site, elle est créée à partir des données de Pamplémousse via un script Python (load_data.py) qui remplit la base avec les 214 élèves de troisième année (et génère les caractéristiques au hasard).

La connexion et les appels à la table 'users' sont faits grâce à la classe *PDO* (PHP Data Objects) en début de page PHP. La variable de session "pseudo" permet d'accéder à l'ensemble des caractéristiques de l'utilisateur qui s'est connecté au site.

2.1.2 La table 'messages'

La table 'message' comporte les messages envoyés par les utilisateurs. À chaque message, est attribué une 'id' (entier) différent lors de sa mise en base. Un message est codé par ligne avec : le destinataire, l'émetteur, l'objet, la date (timestamp), le texte et un booléen qui prend la valeur TRUE si le message a été lu.

Les connexions à la base se font de la même manière que pour la base 'users'. La variable de session "pseudo" permet de récupérer les messages destinés à l'utilisateur connecté.

2.2 Les pages

Lors de la connexion sur la page index.php, l'utilisateur a deux choix : soit se connecter s'il a déjà un compte, soit en créer un via un formulaire. La connexion au site déclenche la création d'une session qui permet de garder en mémoire le pseudo de l'utilisateur tout au long de la navigation.

Hormis les pages de connexion, toutes les pages du site sont structurées de la même manière :

- un *header* qui comprend la bannière centrale avec le nom du site "Affinity Finder" et les fichiers .css
- le corps de page qui inclut un menu horizontal avec quatre options : profil, chercher des personnes, messagerie et paramètres.
- un bas de page qui comprend les mentions légales et les scripts (en Javascript, nécessaires à l'exécution de certaines fonctionnalités du package Bootstrap).

La page disp.php est la page charnière du site, elle permet d'accéder aux autres services proposés par le site (recherche et messagerie). Par défaut, elle affiche le profil de l'utilisateur. Selon l'attribut dans l'URL, elle permet également d'afficher les profils des autres utilisateurs. Lorsqu'un utilisateur A visite le profil d'un utilisateur B, il voit l'ensemble de ses caractéristiques et peut le contacter via le bouton "Envoyer un message."

La recherche de contact se fait par l'onglet "Chercher des personnes" accessible via le menu horizontal. Elle se structure en deux pages : une première dans laquelle l'utilisateur est invité à choisir la matière du projet pour lequel il souhaite trouver un partenaire et une seconde page qui affiche les résultats de la recherche : un

tableau qui comprend la liste des meilleurs partenaires (voir la section Code et structure du code pour plus de détails) compte tenu de ses caractéristiques sous forme de tableau avec nom, prénom et lien vers le profil.

La messagerie s'articule autour de quatre pages :

- la page boîte de réception (message.php) qui affiche la liste des messages reçus par l'utilisateur sous forme de tableau avec l'émetteur, l'objet du message et l'option "lire le message".
- la page lecturemessage.php qui permet de lire un message (sélectionné dans la boîte de réception) et d'y répondre.
- La page "message envoyés" affiche l'ensemble des messages envoyés par l'utilisateur.
- la page écrire message permet à l'utilisateur d'envoyer et affiche également les contacts récents dans la colonne de droite. Lors de la saisie d'un contact, un script d'auto-complétion (autocomplete.php) permet à l'utilisateur de sélectionner rapidement un contact après la saisie des premières lettres de son pseudo.

2.3 Code et structure du code

La structure du code est assez simple : chaque page comporte les éléments dynamiques qui lui sont propres : interaction avec les bases de données et affichage des informations correspondantes. Nous n'avons pas adopté une structure en MVC (Modèle-Vue-Contrôleur) afin de limiter le nombre de pages du site et d'en simplifier la structure.

L'échange de données entre les pages se fait via les fonctions *_POST*, *_GET* de PHP. Lorsque l'utilisateur interagit avec le site (clic, saisie de texte), la validation de ses actions entraîne la redirection vers une autre page du site où sont traitées les informations correspondantes. Ainsi la page connexion (etape1.php) récupère les informations entrées par l'utilisateur mais c'est la page disp.php qui vérifie que l'utilisateur existe bien et qu'il a entré le bon mot de passe. De la même manière, ce sont les pages écriremessage.php et lecturemessage.php qui permettent à l'utilisateur de rédiger un message mais c'est la page message.php vers laquelle ils sont automatiquement redirigés après validation qui traitent l'information.

2.3.1 La fonction recherche

La fonction recherche calcul la distance euclidienne entre un utilisateur A et un utilisateur B selon leurs ca-

ractéristiques respectives $X_{i,j}$ où X_i est une variable binaire (le fait de coder en Latex par exemple, si A et B savent coder en latex, la rédaction sera facilitée).

On attribue ensuite à chaque utilisateur B un score selon une fonction affine décroissante, le score étant ensuite augmenté selon certains aspects ζ_b de l'utilisateur B comme par exemple le fait que la matière préférée de B soit celle du sujet pour lequel A cherche un partenaire. Ainsi :

$$\text{score}_a(b) = \max_{j \in \text{users}} (\text{dist}(a, j)) - \text{dist}(a, b) + \zeta_b$$

Les résultats sont ensuite envoyés à la partie "Vue" qui affiche les utilisateurs de manière décroissante selon leur score (page `simple_search.php`).

2.3.2 Le chargement de la base utilisateurs

Afin de créer une base d'utilisateurs directement utilisable et afin de travailler sur les interactions entre page Web, base de données et langage de script, nous avons codé une fonction qui permet de charger la base des utilisateurs de Pamplémousse en troisième année directement sur notre site (214 élèves). En s'appuyant sur le code source de la page de Pamplémousse, un script (écrit en Python : `load_table.py`) crée un fichier .csv avec nom, prénom (tirés du code source) et caractéristiques (générées aléatoirement) des utilisateurs de Pamplémousse. Le code PHP (page `load_data.php`) lance le script python, récupère le fichier .csv et le transforme en une table SQL directement utilisable par la suite. La page d'accueil du site (`index.php`) vérifie automatiquement si une base d'utilisateurs existe et charge celle de Pamplémousse le cas échéant.

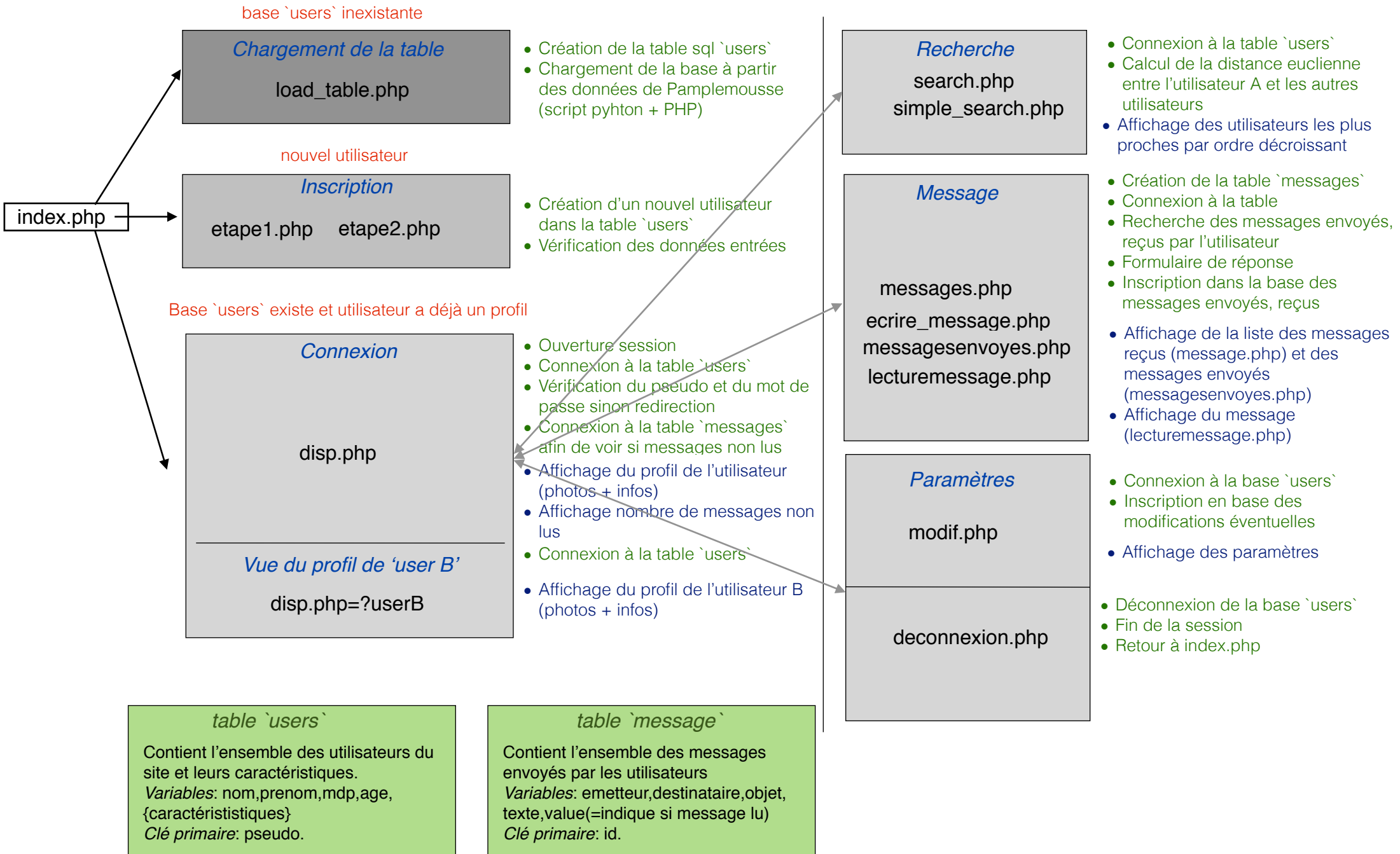
2.4 Conclusion

Ce projet informatique s'est concentré sur l'interaction entre base de données et pages Web. Notre projet met en avant les différents aspects d'une base de données relationnelle : création et lecture de base de données, ajout et modification d'éléments dans une table. À travers les différentes fonctions utilisées et l'usage de script, il a également permis d'utiliser des fonctions usuelles et quelques fonctionnalités avancées qu'offre le langage PHP.

Variable	Type	Caractéristiques
pseudo	varchar(20)	pseudo de l'utilisateur, utilisé comme clé primaire
prenom	varchar(20)	prénom de l'utilisateur
nom	varchar(20)	nom de l'utilisateur
age	int(3)	âge de l'utilisateur
annee	varchar(3)	1A, 2A, cés (césure) ou 3A
voie	varchar(10)	voie choisie en 2A : éco ou maths
matiere	varchar(20)	matière préférée de l'utilisateur
logiciel	varchar(10)	logiciel utilisé pour faire des statistiques
latex	tinyint(2)	l'utilisateur sait-il coder en Latex ?
dents	tinyint(2)	l'utilisateur se lave-t-il les dents ?
timing	tinyint(2)	comment l'utilisateur gère-t-il la DL des projets, plutôt à l'avance ou à la dernière minute ?
anglais	tinyint(2)	l'utilisateur sait-il parler anglais couramment ?

TABLE 1 – La table 'users'

ANNEXE - SCHEMA DU SITE



ANNEXE - SCREENSHOTS

La page de connexion

~ Affinity Finder ~

Se connecter

☐ Se souvenir de moi


Créer un nouveau compte

Copyright , tous droits reserves

Le profil

~ Affinity Finder ~

[Votre Profil](#) [Chercher des personnes](#) [Messages 1](#) [Paramètres](#)



Laura BERTHET

Détails

Année: 3

Vole: eco

Copyright , tous droits reserves

La fonction recherche

~ Affinity Finder ~

[Votre Profil](#) [Chercher des personnes](#) [Messages](#) [Paramètres](#)

Type de sujet

Informatique

Résultats de la Recherche Simple

Prénom	Nom	Score	Profil
Vincent	GENTET	5	voir le profil
Emma	GONDRAN	5	voir le profil
Kévin	VU	5	voir le profil
Nicolas	HUCHET	5	voir le profil
Romain	DAVID	4	voir le profil
Arnaud	DE MYTENAERE	4	voir le profil
Quentin	PROST	4	voir le profil
Benjamin	KOHL	4	voir le profil
Geoffrey	DEVOST	4	voir le profil

Copyright , tous droits reserves

Le profil d'un autre utilisateur

~ Affinity Finder ~

[Votre Profil](#) [Chercher des personnes](#) [Paramètres](#)



Vincent GENTET

Détails

Année:	3
Voie:	eco
Matière préférée:	topologie
Sait coder en Latex :	non
Se lave les dents :	oui
Sait parler en anglais :	oui
Timing :	Plutôt à l'avance

Envoyer un message

Copyright , tous droits reserves

La fonction message

~ Affinity Finder ~

[Votre Profil](#) [Chercher des personnes](#) [Messages](#) [Paramètres](#)

MESSAGES

Nouveau message

Boite de Reception

Envoyes

louis

Projet Économie

Salut !

Je cherche un partenaire pour le projet d'économie.
Veux-tu que nous travaillons ensemble ?

Laura

ENSAE ParisTech

Envoyer

Contacts récents

vincent1

Copyright , tous droits reserves