PROJET BDW1

Guth Romaric
Martinet Aurélien

RAPPEL DU README :

Pour vous connecter et tester les différentes fonctionnalités implémentées utilisez les log suivants :
Profil administrateur : Identifiant : admin , mot de passe : romaric
Profil adhérent : Identifiant : Dupont , mot de passe : Alice

1) Première conception du Modèle

A. SCHÉMA

Nous avons commencé notre réflexion en déterminant les données à stocker dans notre base et leur organisation en tables en adéquation avec les instructions du sujet. Nous avons ensuite réfléchi aux liens devant exister entre les différentes tables pour pouvoir obtenir toutes les informations nécessaires grâce à une requête. Le premier schéma que nous avons obtenu est disponible en annexe (voir (1)).
Nous avons opté pour les liaisons qui nous semblaient les plus logiques :

- L'adhérent est relié à son adresse et ses résultats.
- Les résultats sont liés à une édition et le détail des temps de passages
- Une édition est reliée à une course et une adresse
- Une course est reliée à plusieurs épreuves avec un tarif dépendant de l'âge pour chacune de ces épreuves

B. MODÈLE RELATIONNEL

A partir de ce schéma, nous avons donc pu déterminer les clefs à définir dans nos différentes tables pour pouvoir effectuer ces liaisons. Pour créer ce schéma, nous avons utilisé l'application web mocodo qui nous a généré un modèle, que nous avons réadapté pour obtenir le modèle suivant :

COURSE (idc, nom, mois, url, *anneeCrea*)
EDITION (ided, #idc, annee, #idadr, nbPart, plan, debI, limI, debC, finC, debD, finD)
EPREUVE (idep, dist, deniv, type)
TARIFS (age, #ided, #idep, prix)
ADHERENT (idA, #iidadr, nom, prenom, age, sexe, dateCertif, club)
ADRESSE (idadr, rue, codePost, ville)
RESULTAT (dossard, #ided, rang, nom, prenom, sexe)
TpsPASSAGE (#ided, #dossard, km, temps)

C. CONTRAINTES

Nous avons eu du mal à trouver une manière de représenter les tarifs d'une édition. En effet, ceux-ci dépendaient de l'âge, d'une épreuve et d'une édition et il fallait donc pouvoir les relier à toutes ces informations. Nous nous sommes ensuite rendu compte que les tarifs pouvaient permettre et donc s'insérer dans le lien entre une épreuve et une course, qui était obligatoire.
Nous avons également eu du mal à organiser toutes les données devant être représentées dans la base. En effet, nous ne savions pas toujours s'il fallait créer une nouvelle table pour une certaine donnée ou si nous pouvions la représenter directement dans une table parente.
Nous avons choisi premièrement de définir pour les éditions les dates de début et de fin pour chaque échéance (dépôt des certificats...) mais nous avons envisagé de créer une table représentant en même temps la date de début et celle de fin, mais cela compliquait beaucoup de requêtes.
Nous nous sommes également rendu compte en consultant les données fournies que les dossards n'étaient pas uniques indépendamment des éditions. Nous avons donc dû rajouter l'identifiant de l'édition correspondante à ces données pour former les tables stockant les résultats.
Enfin, nous avons dû attribuer une clef primaire aux adresses pour pouvoir les relier à un adhérent ou bien à une édition selon nos besoins, ce qui n'était pas possible avec une clé étrangère puisqu'elle n'aurait pas pu référencer soit l'identifiant d'une édition, soit celui d'un adhérent.

2) Peuplement de la base

Suite à cela, nous avons dû peupler notre base avec les données proposées par le corps enseignant de l'UE, tâche qui a été relativement laborieuse, car les données fournies ne correspondaient pas forcément au modèle que nous avions (les adresses étaient présentent sous forme de chaines de caractères alors que nous avions créé une nouvelle base à cet effet, guillemets manquants, ect..). Nous avons donc ajusté les données fournies et peuplé les autres tables en fonction de ces données (épreuves, tarifs, …).

3) Création de l'application WEB

A. Authentification

Après avoir peuplé base, nous nous sommes penché sur l'authentification avec deux types de profils (admin et adhérent) que devait gérer notre application WEB. Pour se faire, nous avons choisi de créer une nouvelle table que voici :

IDENTIFICATION (identifiant, #ida, statut, mdp)

Le champ « statut » représentant le « titre » de l'utilisateur, à savoir admin ou adhérent.

Même si selon les spécifications du sujet, un adhérent n'est pas administrateur, nous avons préféré donner tout de même un ida aux administrateurs et les ajouter à la table adhérent pour éviter d'avoir des valeurs « null » à gérer.
Ainsi, à l'arrivée sur le site, l'utilisateur renseigne ses identifiants et l'application peut vérifier leur appartenance à la table identification, puis avoir accès au statut de l'utilisateur et son identifiant d'adhérent.
Nous avons ajouté à la table identification deux sets d'identifiants pour permettre les tests en profil adhérent et administrateur. Par ailleurs, l'administrateur doit attribuer des identifiants à un nouvel adhérent lors de son ajout directement dans l'application.

B. Accueil sur le site

Après l'authentification, l'application redirige l'utilisateur sur son espace personnel qui s'adapte en fonction du statut de l'utilisateur.

- Si l'utilisateur est un adhérent, il a accès à son accueil personnalisé, avec un menu lui permettant de visualiser ses différentes informations.
- Si l'utilisateur est un administrateur, il a accès à une page d'accueil lui permettant la gestion de l'ensemble des données de l'application.

4) Profil adhérent

A. Fonctionnalités

Comme demandé dans le sujet, un adhérent a surtout un rôle de « consultant » sur le site. En effet, excepté la modification de ses données personnelles, il pourra seulement consulter les données de la base tel que les courses, éditions, résultats, et les informations le concernant.

Nous avons donc pour cela respecté le modèle proposé par le sujet, et avons créé les fichiers demandés dont le contenu est ajouté à la demande à l'espace personnel de l'adhérent grâce à un menu, ainsi que les requêtes nécessaires à l'affichage des informations adéquates.

B. Code et difficultés

L'application est gérée comme conseillé sur le sujet : les informations à afficher sont transmises à l'application via l'URL à chaque rafraichissement de la page. Ainsi, un clic sur le menu met à jour l'url en modifiant le champ page de l'URL. Ensuite, le fichier correspondant est inclus ce qui permet d'afficher le contenu HTML qu'il envoie. La modification des données se fait via un formulaire envoyant les données à un script (modif.php) qui exécute la requête permettant la modification dans la base de donnée des champs renseignés. Les résultats des autres requêtes sont affichés dans un tableau grâce à la fonction Array2Table proposée dans le sujet.

Nous avons fait face à quelques difficultés avant de pouvoir parvenir à ce résultat. En effet, la création de la requête de modification était difficile à effectuer car nous voulions que l'application ne modifie pas les champs non renseignés dans le formulaire. Nous avons donc choisi de construire la requête au fur et à mesure en testant la valeur des champs, ceci demandant des concaténations de chaînes de caractères et rendant donc la gestion des « quottes » compliquée (et le débogage tout autant pour trouver ou il manquait un guillemet).

5) Profil administrateur

A. Fonctionnalités

De même que pour le profil adhérent, nous sommes restés relativement fidèles au modèle proposé par le sujet pour le profil administrateur. Celui-ci fonctionne comme le profil précédant, mis à part qu'un administrateur peut exercer beaucoup plus d'action sur la base de données qu'un adhérent. En effet il peut modifier, ajouter ou supprimer des informations concernant les courses, les éditions et les adhérents et bien sûr voir ces informations.

B. Code et difficultés

Le code est également organisé de la même manière que pour le profil adhérent. Cependant, le nombre important de données à gérer a rendu son écriture plus complexe que pour le profil adhérent.
Il y a donc plusieurs scripts permettant l'ajout et la modification des données. Dans les fichiers gérant les courses, nous avons de choisi de passer dans l'url la lettre E ou C suivi d'un identifiant pour toujours savoir les informations de quelle course ou édition il fallait afficher. De plus, nous avons utilisé des vues pour rendre les requêtes plus faciles à écrire et plus légères dans les fichiers php.

Pour l'ajout et la modification, le fait que les données soient réparties sur plusieurs tables nous à encore compliqué la tâche. Nous avons par exemple du utiliser des requêtes multiples avec LAST_INSERT_ID() pour récupérer les identifiants insérés automatiquement dans une table et les insérer dans une autre table en tant que clé étrangère.
Nous avons également choisi d'insérer le formulaire de modification dans un tableau et de remplir les champs par défaut avec la valeur actuelle étant dans la base. Pour ce faire, nous avons utilisé des requêtes renvoyant les champs d'un formulaire grâce à la fonction SQL CONCAT. Nous nous sommes par la suite rendu compte que mettre un valeur vide à un champ d'un formulaire empêchait la création de ce champ. Il a donc fallu tester à chaque fois s'il y avait une valeur dans la table pour l'afficher si c'était le cas, ceci rendant les requêtes encore plus longues et complexes (et toujours avec ces fameuses « quottes »). CONCAT nous à également permis de générer des liens supprimer en passant l'identifiant de la donnée à supprimer.

Encore une fois, c'est donc la gestion de la présence de valeur ou non dans la base qui a été compliquée. Nous avons également défini les valeurs à renseigner obligatoirement lors de la création d'un tuple et avons fait en sorte de permettre également de laisser des champs vides.
Enfin, malheureusement les fonctionnalités en rapport avec des fichiers ne marchent pas sur l'application. Le problème semble être au niveau de la récupération du fichier mais n'avons pas trouvé de solution.

6) Conclusion

Pour conclure, nous estimons avoir respecté la plupart des consignes données dans le sujet, bien que nous ayons dû apporter parfois quelques modifications.
Le souci principal que nous avons rencontré a été le temps. En effet, malgré la semaine supplémentaire qui nous a été accordé, nous avons eu du mal à tenir les délais et à rendre quelque chose de fonctionnel dans les temps. Nous regrettons également d'avoir perdu du temps en devant recommencer des tâches, car nous nous rendions compte en avançant dans le sujet que notre version n'était pas viable ou que le sujet demandait une manière de faire précise. En effet, certaines informations sur les différentes tables demandées sont explicitées très tardivement dans le sujet, par exemple, la table adhérent nous est spécifiée dans le premier paragraphe, mais il nous est demandé plus tard de faire en sorte que l'adhérent puisse modifier son adresse mail, donnée qui n'est pas demandée au début. Nous avons sûrement dû omettre certaines caractéristiques à cause de la grande densité du sujet, que nous n'avons pas forcément réussi à bien gérer dans les temps qui nous étaient impartis
En revanche, cet exercice nous a permis de progresser et d'acquérir de l'expérience dans le domaine du WEB, que nous avions peu pratiqué jusque-là excepté en IHM et en LIFAP5. Ce projet nous a également permis d'apprendre à gérer plus précisément une base de donnée parallèlement à une application WEB.

ANNEXE

(1) Schéma E/A