ASI 02 - Groupe 1

Projet TechnoWeb

Réalisation d'un site web dynamique dédié au diagnostic de territoire

Rapport Global

25 juin 2003

Table des matières

Chapitre 1

Introduction

1.1 Le client : l'ENGREF

L'Ecole Nationale du Génie Rural des Eaux et des Forêts (ENGREF) dispense la Formation des Ingénieurs Forestiers (FIF), est habilitée à délivrer le doctorat dans ses domaines de compétences et développe des collaborations scientifiques avec de nombreux partenaires extérieurs.

1.2 Le mastère SILAT

Le Mastère SILAT (Systèmes d'Informations Localisées pour l'Aménagement des Territoires) permet à des spécialistes thématiques (agronomes, forestiers, hydrologues, environnementalistes) d'acquérir une compétence technique de haut niveau dans le domaine de l'information géographique numérique (géomatique). Outre l'acquisition de solides bases théoriques, la formation vise à donner une très bonne maîtrise pratique des outils et des méthodes, en faisant une large place à la manipulation des données, à la pratique des logiciels, à la réalisation effective de "micro-projets", et surtout à la conduite d'un projet professionnel en entreprise.

1.3 Le projet SILAT

Dans le cadre de ce mastère, Cédric Mahé, sous la direction de Sylvie Lardon, a élaboré un prototype de site web présentant et retraçant l'élaboration d'un diagnostic de territoire. Ce prototype a ensuite été confié au département ASI de l'INSA de Rouen afin qu'il développe un outil pédagogique pleinement opérationnel.

Chapitre 2

Coordination - TT3

2.1 Points clés de notre démarche

2.1.1 Approche d'un projet à 24

Gérer un projet de 24 personnes environ nécessite de l'organisation, et surtout une bonne répartition des tâches. Nous reviendrons sur ce dernier point en partie **??**. Notre objectif était de gérer les groupes en TechnoWeb comme des modules, qu'il suffit d'assembler pour que le site fonctionne. Chaque module a la charge d'une partie du site, qu'il doit concevoir grâce aux éléments mis à sa disposition par le groupe respectif en BD.

Une procédure de test est demandée à chaque module pour vérifier si sa partie est fonctionnelle. Si jamais il ne possède pas les méthodes d'accès à la BDD, il crée un retour factice de la base, qu'il pourra manipuler en attendant la connexion réelle à la base.

Les modules comportent bien évidemment des parties à coder qui se recoupent. L'outil mis en place pour gérer ces mises-à-jour permanentes fut le serveur CVS. Les versions dépassant 100 pour de nombreux fichiers, l'aide du serveur CVS fut très appréciable.

2.1.2 Consignes de codage

Le but de ces consignes était d'uniformiser le code au sein de ce projet afin de faciliter la relecture et l'échange de fichiers entre les différents groupes, mais aussi de rappeler des problèmes couramment rencontrés ainsi que les solutions qui leur sont associées.

Les consignes de codage que nous avons établis sont très largement ins-

pirées de celles fournies par PEAR (`http://pear.php.net/`). Divers documents nous ont également permis de venir compléter ces consignes (pour plus de détails voir la rubrique "sources" du document). De plus, des recommandations propres au projet SILAT sont venues s'ajouter à cela.

Ces consignes portent donc sur tout ce qui concerne la syntaxe (indentation, appel de fonction, structure de contrôle, ...) mais aussi sur différentes méthodes de programmation en PHP ("magic number", guillemets simples ou doubles, ...). Elles regroupent également les décisions prises concernant le projet (package, en-tête des fichiers, ...)

2.1.3 Génération automatique de documentation

Pour la génération automatique de documentation, notre choix devait se faire entre Doxygen et PHPdoc.

Doxygen est un utilitaire qui permet de générer la documentation d'un programme écrit en C, C++, Java, IDL. De plus, il peut être adaptée pour traiter du code PHP. Il est très rapide et relativement puissant. Toutefois, il s'agit d'une application à part entière et il n'est pas directement conçu pour traiter du PHP.

PHPdoc est un ensemble de fichiers PHP capables de générer une documentation. Il ne nécessite donc pas l'installation d'un nouveau logiciel. Bien que légèrement plus long que Doxygen, sa seule vocation est le PHP. De plus, les consignes de PEAR sont en accord avec cette API. C'est donc cette solution que nous avons retenue.

2.1.4 Choix à effectuer

Nous avons à plusieurs reprises été confrontés à des problèmes qui comportaient deux solutions disctinctes, sans qu'il soit possible de faire un choix évident. En voici 2 exemples, et comment nous les avons résolus :

Frame or not frame ?

Globalement, les frames dans les navigateurs les plus répandus actuellement constituent un support non-ergonomique :

- problème d'impression (quel cadre imprimer ? tous ?)
- problème de navigation dans le site (page précédente ... mais de quelle frame ?)
- problème d'inconfort dans la navigation si on a plusieurs ascenseurs sur une même page

– augmente l'instabilité de la mise en forme : d'une augmentation de la taille des polices peut résulter l'apparition de multiples barres de défilement
– impossibilité de bookmarker un frameset correctement
– les résultats des moteurs de recherche renvoient l'url d'une frame sans son frameset : cela rend plus complexe l'accès aux autres informations contenues sur le site

Cependant l'utilisation d'une page unique composée de tableaux "simulant" les frames est lourde à gérer. D'une part, la gestion des login et l'interface sécurisée utilise des "header" qui ne permettent pas l'envoi de code avant cette fonction. Il faut donc placer ce code en début de page. Sur une page sans frame, cela nécessite d'insérer ce code en haut de la page, et le code de traitement et d'affichage plus en dessous. Il aurait fallu scinder de nombreux fichiers en 2 ou 3 parties pour gérer ne serait-ce que l'affichage.

D'autre part, notre gestion des groupes se faisant par modules, nous voulions pouvoir insérer chaque partie de site réalisée dans notre page principale, en y accédant indépendamment. Cela est plus pratique au niveau des phases de tests.

Pour conclure, l'utilisation d'une page unique aurait été plus professionnelle, et pu être techniquement réalisable avec plus de temps disponible. Mais dans un esprit de simplicité et de concentration sur les éléments plus fondamentaux (site fonctionnel), nous avons adopté l'usage du frameset.

Dimensions du site

Entre l'affichage du document en cours de visualisation, l'interface et la carte de positionnement, nous avions beaucoup d'éléments à afficher simultanément. Nous nous sommes donc demandé s'il fallait conserver la résolution de la maquette réalisée par Cédric Mahé (800x600) ou si on adoptait une résolution plus confortable de 1024x768. La question a été soumise à Sylvie Lardon : Elle a nous a demandé de laisser le site en 800x600 afin que les étudiants puissent utiliser l'application sur la plupart des ordinateurs de l'ENGREF.

2.1.5 Réunions

L'objectif des réunions était simple : garder la cohérence du groupe. Il faut dans la mesure du possible évoquer les questions générales en groupe. D'une part, le problème que se pose un groupe est peut-être déjà résolu par un autre, ou sera rencontré par un autre groupe. D'autre part, le fait de discuter du projet amène les gens à employer les termes du projet, et permet

de mettre à nu les points incompris par certains. Ce n'est pas infaillible mais lorsque l'on couvre plusieurs sujets, la compréhension globale est bonne.

Voici comment nous procédions : avant chaque réunion, nous recuillions les questions des divers groupes. Si la question était simple, nous y répondions, et l'évoquions ou non durant la réunion comme problème résolu. Dans le cas où une discussion avec d'autres groupes était nécessaire, elle était indiquée à l'ordre du jour. Cet ordre du jour est affiché sur la page du TT3 sur le site du projet SILAT. Il était également inclus dans les mails signalant les réunions à venir.

Un représentant par TT (voire deux) venait à chaque réunion, même s'il n'avait pas de question. Un membre du TT de coordination prenait la parole pour diriger la réunion selon l'ordre du jour. Une seconde personne prenait des notes. Les sujets étaient donc passés en revue, en donnant la parole à chaque groupe de TT. A chaque fin de réunion un tour de table était proposé pour évoquer les diverses questions restantes.

Suite à une réunion, un compte rendu est rédigé, évoquant les problèmes, les solutions apportées et les arguments de chacun des parties. Ils sont disponibles sur CVS, dans le répertoire "CR". Ils sont également inclus en annexes (Partie **??**), car ils sont témoins des décisions prises, et à quelle date.

2.2 Problèmes rencontrés

2.2.1 Evaluation du travail à effectuer et répartition des groupes

Dans la partie TechnoWeb du projet SILAT, le découpage et la répartition des différentes parties ont souffert d'une mauvaise évaluation de la quantité de travail nécessaire. En effet, ce projet constituait pour beaucoup d'entre nous, le premier contact avec un projet de cette envergure.

De plus, dans tout projet, il est toujours difficile de prévoir à l'avance le temps qui sera occupé pour résoudre les différents problèmes rencontrés. Ceci explique que certaines des parties, définies pendant les premières semaines, aient été plus longues que les autres et donc les différences d'avancement qui peuvent exister.

Enfin, certaines parties du projet étaient totalement dépendantes des autres et ne pouvaient donc être développées, ou du moins testées, avant que le travail ne soit terminé sur les fichiers nécessaires. Ces divers éléments expliquent la surcharge de travail qui est apparue chez certains groupes à la fin du projet.

2.2.2 Communication avec l'ENGREF

Notre interlocutrice principale était Sylvie Lardon, de l'ENGREF de Clermont-Ferrand. Notre principal support de communication a été l'e-mail, mais nous avons aussi utilisé la salle de visio-conférence de l'INSA. Michel Mainguenaud a également été un relais entre les étudiants et l'ENGREF.

Il est indéniable que la communication entre l'ENGREF et les étudiants en charge du projet n'a pas été parfaite. Cependant, l'éloignement géographique entre l'INSA de Rouen et L'ENGREF ne semble pas en être la cause. Il s'agit surtout d'une différence de "culture" : nous avons déjà entre étudiants des problèmes à nous comprendre - entre étudiants ou entre étudiants et enseignants - de temps en temps, il est donc logique que ces problèmes soient d'autant plus importants avec des gens dont l'informatique n'est pas la spécialité.

2.2.3 Compréhension du sujet

Une certaine période de prise de contact, plus longue que prévue, a été nécessaire pour nous permettre de nous familiariser avec le sujet. Tout d'abord, il nous a fallu comprendre le fonctionnement du logiciel existant, ainsi que les nouvelles attentes de nos clients. Ensuite, nous avons du commencer à spécifier les différentes fonctions nécessaires ce qui nous a permis de nous poser des questions et de mieux voir les problèmes sous-jacents. Finalement, nous avons du mettre en place les solutions trouvées ce qui nous a permis de mieux appréhender le langage PHP.

2.2.4 Ecarts entre spécifications et développement

Dans un développement idéal, conforme à la théorie, le déroulement normal veut que l'on établisse d'abord les spécifications puis que le codage ne soit que la transcription dans le langage choisi de ces spécifications. C'est pourquoi, la règle veut que la partie de spécification représente 80% du temps consacré à un projet, le reste étant consacré au codage. Toutefois, un projet n'est jamais un cas idéal, et malgré le temps consacré aux spécifications, des oublis ou des erreurs sont toujours possibles. De plus, on peut être amené à remettre en cause les spécifications lorsque celles-ci sont irréalisables dans le langage choisi. Ainsi, dans la plupart des projets, quelques "navettes" entre les spécifications et le codage peuvent être nécessaires avant d'arriver à la version définitive du projet. Dans le projet, c'est ce temps de perfectionnement qui nous a manqué. En effet, les spécifications ont bel et bien pris 80% du temps, donc il aurait fallu un codage parfait pour respecter les

délais. Hors, des problèmes d'accès aux informations se sont posés du fait de la spécification des classes. En effet, en voulant au maximum minimiser la redondance de l'information, cela a complexifié le chemin d'accès au travers des classes jusqu'à rendre parfois impossible la récupération de l'information requise. Quelques modifications au niveau de la répartition de l'information dans les classes auraient du être effectué. Cependant, ces modifications entraîneraient de nombreux changements dans les fichiers déjà existants. Nous n'avons donc pas pris la décision de procéder à ces modifications du fait du manque de temps.

Chapitre 3

Login, sessions - TT1

3.1 Introduction

Concernant la partie Technoweb, nous avions la charge de la gestion des logs et des comptes.

Ceci regroupe trois étapes distinctes :

Tout d'abord, nous devions concevoir et implémenter les fonctionnalités permettant à un utilisateur de se connecter. Comme convenu au début du projet avec les autres TT, ces fonctionnalités devaient être organisées dans des classes, celles-ci devant être implémentées en PHP. Chaque groupe devait également concevoir les pages qui correspondaient à ses fonctionnalités.

En second point, nous devions concevoir les fonctionnalités et les pages pour l'administrateur, à savoir créer un compte ou en supprimer un par exemple.

Pour finir, nous devions concevoir et implémenter des fonctionnalités permettant de sécuriser le site Silat. Pour être plus précis, nous devions concevoir un système de façon à ce qu'un utilisateur soit obligé de se connecter pour accéder aux différentes pages et que chaque groupe d'utilisateur ne puisse accéder qu'à ses pages réservées.

3.2 Classe utilisée pour la gestion des comptes

3.2.1 Classe utilisée

Contrairement à l'ensemble des autres TT, nous n'avons eu à gérer qu'une seule classe pour la partie Technoweb : la classe Utilisateur. De plus, c'est une classe dont seuls deux groupes se servent, à savoir que nous avons implémenté 19 des 21 fonctionnalités de cette classe du fait de nos besoins.

3.2.2 Intégration de la classe Utilisateur dans la diagramme de classes général

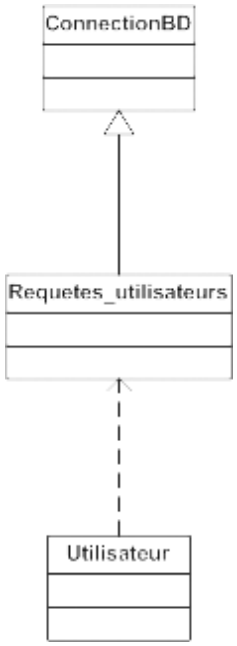

FIG. 3.1 – Diagramme de classe concernant la partie Gestion des comptes dans le projet global

3.2.3 Description de la classe Utilisateur

Elle a pour unique but de gérer le compte d'un utilisateur et les informations relatives à celui-ci.

La classe Utilisateur possède donc des fonctionnalités permettant de traiter à volonté le compte d'un utilisateur. Elle permet également d'utiliser les différentes fonctions de requêtes qui accèdent à la base. C'est en fait l'encapsulation des fonctions créées dans la partie base de données.

La classe Utilisateur n'est reliée à aucune autre classe et aucune classe ne l'utilise. Elle depend seulement de la classe Requetes_utilisateurs qui a été créée dans la partie Base de données du projet Silat.

3.2.4 Fonctionnalités associées

Les fonctions de la classe Utilisateur se découpe en cinq catégories en plus du constructeur :

Tout d'abord, il y a les fonctions d'insertion et de suppression d'un compte dans la base de données :
- ajouterCompte
- entrerInformationsCompte
- supprimerCompte

Ces fonctions permettent de créer ou de supprimer un compte de la base. Afin de modulariser au mieux nos fonctions, nous avons préféré faire deux fonctions pour entrer les informations d'un compte nouvellement créé. En premier lieu, nous devons rentrer uniquement le login qui est la véritable signature d'un utilisateur. Ensuite, nous rentrons les différentes informations alphanumériques qui le concernent. Ceci permet qu'à l'avenir, la maintenance soit plus facile. En effet, si l'on veut rajouter des informations sur un utilisateur, il suffit de changer uniquement la fonction entrerInformationsCompte.

Ensuite, il y a les fonctions permettant de modifier les paramètres d'un utilisateur :
- modifierMail
- modifierNom
- modifierPrenom
- modifierMotDePasse

Celles-ci vont toutes modifier les informations d'un utilisateur dans la base de données. Nous avons préféré créer une fonction par paramètre de façon à en avoir une utilisation très souple. Par exemple, dans l'état actuel des choses, il existe une page pour changer son nom, son prénom et son mail et une page pour changer son mot de passe. Grâce à la modularité de ces

fonctions, il est possible de changer les pages, comme créer une page par paramètre que l'on veut modifier, sans changer les fonctions.

Ensuite, les fonctions suivantes permettent d'obtenir les paramètres d'un utilisateur :

- obtenirMail
- obtenirNom
- obtenirPrenom
- obtenirMotDePasse
- obtenirLogin
- obtenirListeLoginNomPrenom

Toutes ces fonctions vont permettre d'obtenir les paramètres d'un utilisateur se trouvant dans la base de données. A l'instar des fonctions de modification, nous avons créé une fonction par paramètre de façon à être le plus modulaire possible. Comme pour les fonctions précédentes, cela permet de pouvoir changer les pages sans changer les fonctions.
La fonction obtenirListeLoginNomPrenom est quelque peu différente des autres. Elle permet d'obtenir la liste de tous utilisateurs avec leur login, leur nom et leur prenom. Ceci nous est utile lors du choix de suppression d'un compte : nous pouvons affiché ainsi la liste des comptes (sauf celui de l'administrateur) avec le login, le nom et le prénom.

La quatrième catégorie comporte les fonctions de vérification :

- verificationLoginDejaPresent
- verificationMotDePasseCorrect

Lors de la connexion d'un utilisateur, plusieurs tests sont effectués. Tout d'abord, nous vérifions que les champ du formulaire sont bien remplis. Ensuite nous utilisons ces deux fonctions pour vérifier que le login de l'utilisateur est bien existant et que le mot de passe rentré par celui-ci est correct. La vérification du mot de passe est simple : nous comparons le mot de passe rentré par l'utilisateur avec celui associé au login dans la base de données. Les retours de ces fonctions sont uniquement des booléens.

La dernière catégorie traite des mails :

- remplacerMotDePasseOublie

Cette fonction permet de renvoyer un mail à un utilisateur qui aurait oublié son mot de passe. Celui-ci reçoit ainsi son nouveau mot à l'adresse indiquée lors de la création du compte.

En plus de ces fonctions, nous avons utilisé deux fonctions pour les mots

de passe ne servant que pour nos fonctionnalités :

- crypterMotDePasse
- genererMotDePasse

La première fonction permet de crypter un mot de passe et ainsi de pouvoir en changer très rapidement. A chaque fois qu'un mot est rentré dans la base, celui-ci est crypté auparavant en utilisant cette fonction. Ceci permet qu'il est impossible de retrouver le mot de passe d'un utilisateur à partir de la base de données. Actuellement, nous utilisons l'algorithme md5 qui crypte sur 32 caractères hexadécimaux quel que soit la taille de la chaîne de caractères passée en entrée.
La seconde fonction permet de générer un mot de passe de huit caractères alphanumériques pseudo-aléatoirement. Cette fonction sert uniquement pour l'envoi d'un nouveau mot de passe par mail en cas d'oubli de celui-ci.

3.3 Démarche adoptée pour la conception

A l'instar des autres groupes, nous avons suivi différentes étapes de conception.

3.3.1 Conception de la maquette

Tout d'abord, nous avons fait une maquette pour représenter les pages de notre partie. Celles-ci devaient être le plus explicatif possible pour avoir une idée précise de la navigation dans nos pages. Par exmple, nous devions indiquer vers où pointaient les liens et quelles étaient les fonctionnalités utilisées pour chaque page. C'est durant cette étape que nous avons décidé des options disponibles pour chaque utilisateur (tel que changer de mot de passe par exemple) et du rôle de l'administrateur.

3.3.2 Connexion d'un utilisateur

La connexion d'un utilisateur est simple : celui-ci rentre son login et son mot de passe. L'identification de l'utilisateur est une comparaison du mot de passe associé au login dans la base de données avec celui rentré par l'utilisateur. Ensuite, celui-ci est renvoyé vers la page correspondant à son groupe ou vers la page de connexion en cas d'échec d'identification.

C'est à partir de ce moment que l'on conserve des informations sur l'utilisateur connecté. Ceci permet par la suite à tout instant quel est son groupe et quel est son login.

3.3.3 Options pour un utilisateur

Un utilisateur possède deux options quel que soit son groupe : changer ses informations personnelles (mot de passe, nom, prénom et email) et obtenir un nouveau mot de passe en cas d'oubli. Nous avons choisi de faire apparaître ces deux options dans la même page que l'identification. En effet, si nous ne le disposions pas dans cette partie, cela nous obligeait à les rajouter dans chaque page d'accueil des groupes. Cela impliquait de créer plusieurs fois le même lien vers une page identique pour chacun. De plus, nous avons statué que le changement de nom et de mot de passe devait pouvoir être effectué rapidement, à savoir en l'absence de toute connexion au préalable. Ceci implique de laisser ces options dans la page de connexion.

3.3.4 gestion de l'administrateur

Par la suite, nous avons défini les fonctionnalités disponibles pour l'administrateur. Celui-ci est en charge des comptes, à savoir de la création et de la suppression de ceux-ci. Il est unique et ne peut accéder aux pages des autres groupes. Son rôle s'arrête uniquement à la gestion des comptes.

Etant donné que celui-ci a un rôle important, la création du compte administrateur est la première action à faire après l'installation du site Silat. Pour ceci, il faut aller sur la page de connexion. Si le compte administrateur n'existe pas, une page permettant de le créer apparaitra. C'est l'unique moyen de le créer et c'est la seule fois où cela sera possible. Il est possible de modifier par la suite les informations relatives à l'administrateur comme tout autre compte. Le login de l'administrateur est prédéfini et ne peut être changé, c'est "admin".

3.3.5 Gestion de la sécurité

La gestion de la sécurité est relativement simple.

Tout d'abord, il y a l'inclusion d'un fichier en haut de chaque page qui contient les différentes fonctionnalités nécessaires à celle-ci. La première étape est de vérifier que l'utilisateur est connecté. Si celui-ci ne l'est pas, il est alors renvoyé vers la page d'identification.

Ensuite, le groupe auquel appartient l'utilisateur est comparé avec le groupe associé à la page (par exemple "professeur" pour les pages concernant la création d'un exercice). Si celui-ci n'est pas correct, il est renvoyé

vers la page d'identification.

Ces deux vérifications évitent qu'un utilisateur mal intentionné aille sur certaines pages auxquelles il ne devrait pas avoir accès en tapant directement l'adresse de la page dans son navigateur.

3.3.6 Tests de nos pages

Contrairement aux autres groupes, nous avons pu effectuer des tests sur nos pages très tôt.En effet, nous avions la responsabilité des fonctions relatives aux comptes dans la partie Base de Données. Nous avons donc implémenté au plus vite ces fonctions pour intéragir avec la base de données. Après l'installation de PosgresSQL, d'Apache et de PHP, ceci nous a permi de tester très rapidement nos pages.

D'ailleurs, nous avons pu présenter des pages fonctionnelles et intéragissant avec la base de données lors de la visio-conférence avec l'ENGREF le mercredi 4 juin 2003.

3.4 Problèmes rencontrés

3.4.1 Gestion des sessions

Le premier problème que nous ayons rencontré est la gestion des sessions. En effet, celles-ci sont indispensables pour conserver les informations sur l'utilisateur connecté et ont demandé de nombreux tests. Désormais, c'est un objet Utilisateur qui est conservé dans une variable de session, ce qui permet d'obtenir toutes les informations disponibles sur lui sans se connecter à la base de données.

3.4.2 Changements fréquents des spécifications

Un problème que beaucoup de groupes ont connu également est l'évolution constante des spécifications, ce qui a entraîné à de nombreuses reprises des changements dans la conception des fonctionnalités. Ceci est d'ailleurs plus génant quand le code a été testé et que tout fonctionne correctement.

3.4.3 Fonctions dans les mêmes pages que l'affichage

Après avoir résolu le problème des sessions, nous avons pu faire nos tests avec nos fonctionnalités en créant les différentes pages et en les faisant

intéragir avec la base de données. Cependant, les liens de certaines pages pointaient vers une même page. Par exemple, après avoir changé de mot de passe ou de nom, on arrive sur la même page, à savoir le menu pour choisir quelle information personnelle changer.

Initialement, nous avons procédé avec des variables de session pour différencier de quelle page nous venions, ce qui est vite devenu compliqué et impossible à comprendre pour des personnes n'ayant pas implémenter cette partie. C'est pourquoi nous avons décidé de séparer l'affichage de l'execution des fonctionnalités. Nous avons ainsi créé des pages intermédiaires entre les pages d'affichage. Ces pages contiennent l'appel des fonctions adéquates et au final renvoient sur la page suivante. Cette méthode a permi de simplifier le code et a par la même occasion améliorer sa lisibilité. La maintenance est aussi simplifiée car on sait désormais où sont appelées les fonctions, ce qui permet de gagner beaucoup de temps lors d'éventuels changements.

3.4.4 Inclusions en cascade

Le dernier problème que nous ayons rencontré se trouve dans les multiples inclusions de fichiers. En effet, de par l'arborescence du site Silat, les inclusions dans les différents fichiers ne fonctionnaient pas. Par exemple, dans la classe de requetes que l'on utilise (Requetes_utilisateurs), nous faisions une inclusion de la classe mère ConnectionBD. Ensuite, nous faisions une inclusion de celle-là dans la classe que nous avons implémenté pour la partie Technoweb, à savoir les fonctionnalités utilisées dans les pages.

Afin que tout soit inclus et que tout marche, il faut inclure tous les fichiers dans les pages, c'est à dire ne faire aucune inclusion dans les fichiers des classe.

Chapitre 4

Création d'exercices - TT2

Lors de la présentation du sujet du projet technoweb aucune découpe en parties n'avait été réalisée. C'était donc aux différents groupes de s'entendre sur ce découpage et la répartition de chaque partie.

Nous avions déjà à charge le suivi du projet BD. Cette partie nous demandant de moins en moins de temps, nous nous sommes alors proposés pour avoir une partie où le développement serait un peu plus longue.

Malheureusement, lors de ce découpage, nous avons mal estimé le temps de développement de la partie "création d'exercice".

4.1 Organisation du travail

Pour mener à bien ce projet, nous avons dans un premier temps réalisé une description informelle. Cette description explique de manière générale l'organisation de notre partie. Elle contient les différents groupes interagissant avec notre partie, les points importants à développer, la démarche que l'on va suivre pour correspondre aux mieux aux attentes du client.

Dans un deuxième temps, nous sommes passés à la phase de spécifications. Ce document contient les maquettes et les fonctions en rapport avec les autres groupes. Nous y indiquons la marche à suivre pour la programmation. Il permet au client d'avoir une idée sur la forme de son produit. C'est à partir de ce moment qu'il faut se mettre d'accord avec les autres groupes sur la manière d'aborder les différents problèmes et les solutions apportées.

Enfin, la partie codage. Elle nous a demandé énormément de temps et de patience. Nous n'avons pas pu l'aboutir dû à divers facteurs : langage non adapté et non maîtrisé, partie conséquente.

4.2 Description de la création “d’exerice”

Cette partie permet de créer un exercice qui va être, par la suite, lu par le groupe lecteur. Cette interface est accessible à l’ “enseignant”, pour créer, éditer et gérer les exercices lui appartenant.

Pour ce faire, pendant la phase de spécification, nous nous sommes mis à la place d’un professeur. Nous nous sommes fixés l’objectif de créer une interface intuitive et simple d’utilisation. Cela passe par un cheminement clair et une gestion des erreurs précise. Le plus simple est de faire apparaître les boutons utiles au bon moment. De ce fait, l’utilisateur ne peut pas se tromper sur la démarche à suivre.

4.3 Marche à suivre

Les étapes suivantes présentent la démarche de création d’un exercice.

4.3.1 Arrivé sur la page d’accueil

L’utilisateur arrive sur la page d’accueil de “création d’exercice”. Elle affiche les exercices déjà créés. Il peut les modifier, les supprimer ou les dupliquer. Ou créer un nouvel exercice.

4.3.2 Création d’un exercice

Pour cela il faut entrer le nom de l’exercice et l’auteur.

4.3.3 Création des étapes, des séquences et des documents

Bien entendu, ils doivent être modifiables à souhait. Ils ne peuvent être créés que si l’exercice possède un nom et un auteur. La création d’étapes et de séquences n’est pas très compliquée, il s’agit juste d’entrer des caractéristiques.

Pour la création d’un document, c’est beaucoup plus difficile. Outre des caractéristiques simples à entrer, il faut permettre à l’utilisateur de pouvoir insérer un nouveau document dans la base comme une carte ou un chorème.

L’utilisateur doit pouvoir aussi importer un document de la base, déjà créé par un utilisateur lambda, qui est utilisé dans un autre exercice. Ces deux fonctions ont une interface, mais ne sont pas opérationnelles.

4.3.4 Agencer l'exercice

C'est organiser les séquences dans les étapes, puis les documents dans les séquences.

Dès qu'une séquence ou un document est inséré, il ne doit plus figurer dans la liste de choix de séquence ou document. Il faut aussi penser à pouvoir supprimer les séquences et les documents insérés, mais aussi à pouvoir ordonner les documents, c'est-à-dire déplacer un document vers le haut ou le bas dans la liste des documents appartenant à une séquence. C'est ainsi que l'on définit un ordre entre les documents.

Bien entendu, le choix agencement ne doit être valide qu'à partir du moment où l'utilisateur a au moins créé une étape, une séquence et un document.

4.3.5 Réaliser la gestion des liens

Cela permet de créer un lien typé entre deux documents.

Cette partie n'est pas très évidente puis lorsqu'on crée un lien entre documents d'un certain type, il faut automatiquement créer le lien symétrique avec le type symétrique. Exemple : si on crée le lien suivant entre doc1 et doc2, un lien précédent entre doc2 et doc1 est automatiquement créé.

Il faut aussi tester qu'il n'y a pas de document sans lien. Cette partie doit être valide lorsque l'agencement a été réalisé. Malheureusement, la gestion des liens est loin d'être terminée.

4.3.6 Définir un parcours type

Cela est nécessaire pour le groupe Delta. C'est grâce à cette partie qu'il est possible de faire un suivi de l'élève.

Nous nous sommes très mal entendu avec ce groupe sur le parcours type. Nous pensions qu'un parcours type était basé sur l'importance d'un document, alors qu'en fait c'est sur l'importance d'un lien et sur la théorie des graphes, non développée par ce groupe dans les spécifications. Ceci étant dit, le choix de l'importance d'un document est opérationnel.

En revanche ce parcours type n'est même pas mentionné dans notre rapport de fonctionnalité, étant rajouté dans les derniers moments, nous n'avons pas eu le temps de le développer.

4.3.7 Valider l'exercice pour l'enregistrer dans la base

Après avoir enregistré l'exercice, l'utilisateur doit le finaliser pour le rendre visible pour les lecteurs. Sans cette action, l'exercice ne peut être vu par la partie lecteur. Cette action ne peut être disponible qu'après avoir réalisé les étapes précédentes. Cette fonction n'est pas codée.

4.4 Méthode employée

Nous avons appliqué une méthode simple tout au long de la création de l'exercice par l' "enseignant". Les différentes interfaces et le code derrière celles-ci manipulent un objet PHP "exercice", qui est complété au fur et à mesure, et passé de page en page par une variable de session exercice. Lors de la validation de l'exercice par l' "enseignant" (en fait quand il a fini de travailler, même si l'exercice n'est pas "finalisé"), cet objet est entré dans la base de données du projet, selon l'architecture qui lui est propre (qui est assez différente de l'architecture des objets PHP).

Nous devions donc réaliser un interfaçage entre la partie TW et la partie BD (un "BD wrapper"), mais cette partie n'a pas été réalisée. Il faut noter qu'on appelle "opérationnelles" les fonctions qui complètent l'objet "exercice". C'est-à-dire que c'est fonctions fonctionnent en terme de conception objet mais leur interfaçage avec la BD n'a pas été réalisé. La création d'un exercice est donc pour l'instant "virtuelle", et disparaît quand on ferme le navigateur. Nous n'avions pas encore lié notre objet exercice avec les fonctions de la base.

De plus, nous permettons à l'utilisateur de pouvoir reprendre son exercice à tout moment, c'est pourquoi il nous faut des fonctions qui transforment tous les tuples de la BD qui nous intéressent en objet "exercice" (le même "BD wrapper", mais dans l'autre sens). Pour ce faire, nous pensons qu'une étroite collaboration avec le groupe lecteur pourrait nous permettre de gagner du temps (déjà réalisé par le TT6).

4.5 Récapitulatif de travail effectué

- **Ce qui fonctionne**
  - La création d'un exercice
  - La création d'étape, séquence, et document avec possibilité de modification.
  - L'agencement.

  - L'importance des documents.
- **Ce qui ne fonctionne pas**
  - La gestion des liens
  - Le parcours type
  - Le BD Wrapper (relations avec la base en I/O)
  - La majeure partie des gestions d'erreurs (bien que le système de gestion d'erreur ait été réalisé, il n'est fonctionnel qu'en partie, car il nécessite une certaine finalisation de toute la partie de création, pour pouvoir limiter les erreurs de l'utilisateur)
  - Les fonctions permettant de simplifier la création comme les boutons qui apparaissent sous certaines conditions.
  - La fonction de finalisation qui permet de rendre visible l'exercice pour l'élève.
  - Page d'accueil exercice.

4.6 Conclusion

Nous ne pensons pas qu'il y ait de réelles difficultés sur ce projet. La grosse difficulté est liée au langage de programmation imposé. En effet, nous manipulons que des objets et comme tout bon programmateur qui se respecte le sait, le PHP n'est pas fait pour ce genre de chose. Heureusement pour palier à ce problème M. CONALLEN, un des dieux de l'UML, a inventé quelques astuces pratiques pour manipuler des objets avec des langages de types PHP (merci M. CONALLEN).

Nous avons perdu énormément de temps sur l'apprentissage du langage et les parades pour pouvoir manipuler nos objets comme nous le désirons. Le PHP n'a pas de bon débuggeur, de ce fait nous avons passé certaines fois plus d'une heure de suite à trouver la source d'un bug.

Il serait utile et nécessaire de mettre dans le cours de PHP des fonctions comme "print_R", qui permet d'afficher le contenu d'une variable quelque soit son type. Par exemple, pour un objet, on aperçoit tout ce qu'il contient. Ce genre de petits désagréments nous a coûté du temps, ralenti notre avancement et donc empêché de finir notre projet.

Nous ne perdons pas de vue que ce projet est très intéressant et très enrichissant sur le plan personnel mais le problème est la mauvaise estimation du temps et le partage des différentes parties.

Nous finirons sur une note positive : notre TT ayant une partie importante dans la réalisation de ce projet, il va de soit que l'un d'entre nous doit le reprendre dans le cas d'une UV libre. C'est pourquoi Thomas

se propose pour le finir dès la rentrée septembre.

Chapitre 5

Consultation - TT6

5.1 Introduction

Notre objectif était de fournir toutes les fonctionnalités nécessaires au parcours d'un exercice. Cette partie est donc très importante et à necessité une cohérence totale avec quasiement tous les groupes (création de la carte, création d'un exercice, suivi).

Le travail c'est relativement bien passé, la cohérence avec le tt4 qui réalisait la partie lecteur au biveau BD n'a pas posé de problème. Nos fonctions coincidaient totalement, ils ont bien répondu à nos attentes.

5.2 L'organisation du travail

Pour les spécifications, nous avons réalisé un modèle Latex (que le tt3 à complété par la suite), qui a normalement servi à tous les groupes, ce qui simplifiait la lecture des spécifiactions des autres.

D'autre part, nous étions en charge de la cohérence au niveau BD. Nous avons donc veillé à ce que les variables et le diagramme des classes soit cohérent par rapport au schéma de la base, ce qui a largement simplifié les réunions et le codage.

Enfin pour le développement, l'un de nous était root sur le serveur qui nous était alloué. Nous avons donc mis en place avec le TT2 et un groupe du projet A4IR un serveur web apache, PHP, ming (pour le flash) ainsi qu'un serveur CVS, le tout sur une debian stable, ce qui a posé quelques problème car la version de php sur cette distribution est un petit peu ancienne et beaucoup d"évolution ont eu lieu, surtout au niveau de la gestion de Postgres.

Le codage a donc été entièrement géré via CVS. Un script executé par une interface web nous permettait de voir en direct notre travail. On pouvait donc réellement travailler tous en même temsp sur le projet, nos modifications étaient prises en compte immédiatement.

Quand à l'organisation au sein du TT, nous nous sommes répartit les réunions et avons discuté ensemble de nos spécifications (diagramme de classes et maquettes essentiellement). Nous avons eu la même logique avec le codage, en se répartissant les maquettes.

5.3 Les fonctions requises pour Lecteur

5.3.1 Dans le diagramme de classes

- Exercice
  - initExercice4Lecture($_numero_exercice, $_version)
    Cette fonction permet de charger a partir de la base toutes les infos necessaire à l'exercice
  - obtenirNumero()
  - obtenirId_graphe4exercice()
    Cette fonction renvoi la valeur de id_graphe pour le login exercice. Cette valeur va permettre de trouver les liens entre les documents pour cet exercice.
  - obtenirNom()
  - obtenirLogin()
  - obtenirVersion()
  - obtenirListeExercices()
    Retourne un tableau qui contient tous les exercices.
- Sequence
  - initSequence4Lecture($_numero_exercice, $_version, $_numero_sequence)
    Cette fonction permet de charger a partir de la base toutes les infos necessaire à une séquence
  - obtenirDocuments()
    renvoi un tableau qui contient tous les numero de noeuds de documnts contenu dans la séquence.
  - obtenirQuestion()
  - obtenirNom()
  - obtenirNumero()
- Document
  - initDocument4Lecture($_numero_exercice, $_version, $_numero_document, $_id_graphe4exercice)

Cette fonction permet de charger a partir de la base toutes les infos necessaire à un document. On voit ici apparaitre la variable id_graphe initialisée dans la classe Exercice ; nous en avons besoins pour trouver les liens du document.
  - obtenirNumero()
  - obtenirAttribut()
  - obtenirLiens()
    Renvoi un tableau qui contient la deuxieme partie de la clef primaire des liens (en réalité l'ordre). La première partie étant id_graphe.
  - obtenirNom()
  - obtenirCommentaire()
  - obtenirAuteur()
- Attribut
  - initAttribut4Lecture($_id_theme, $_id_typedoc, $_id_typeref)
    Cette fonction permet de charger a partir de la base toutes les infos necessaire à un attribut
  - obtenirTypeRef()
  - obtenirTypeDoc()
  - obtenirTheme()
- Utilisateur
  - obtenirLogin()

5.3.2 Classe spécifique

La page qui permet l'affichage du document necessite de regrouper les liens suivant leur type et la séquence où ils pointent. Le diagramme de classes prévu dans les spécifications rendait difficile cette approche. Nous avons donc créé une classe spécifique (afficheDocument) qui contient une fonction permettant de créer une sorte de vue des liens à partir du diagramme des classes prévu dans les spécifications.

Elle contient la fonction :

```
obtenirSequencesLiees(\$\_id\_graphe, \$\_ordres,
\$\_numero_exercice, \$\_version, \$\_numero\_noeud)
```

Qui renvoi un tableau affichable par la page affiche_document.php

5.4 les problèmes rencontrés

Il y a eu une grosse confusion au début du dévelopement car nous ne suivions pas assez le diagramme de classes. Il en résultait un code peu portable,

qui empiétait sur les parties d'autres groupes. Nous avons donc du refaire un départ. Ceci n'est pas forcement un mal, car nous pouvons considérer la premiere approche de développement comme un essai, une maquette, qui a permi d'obtenir un code beaucoup plus simple et propre par la suite. L'élaboration des maquettes fu périlleuse car il est difficile de s'imaginer au milieu des classes les fonctions dont nous auront besoins.

5.5 conclusion

Ce projet était interréssant mais aurait nécessité un temps de développement plus grand. En fait nous pensons que des maquettes fonctionnelles auraient rendu le codage plus simple et auraient permi un diagramme des classes plus fin.

C'est la première fois que nous avons réalisé un projet ou beaucoup de personnes travaillaient sur les mêmes fichiers. Il est surprenant quand on développe de voir la taille des fichiers grossir de fonctions que l'on ne connait pas, de les utiliser, et finalement à la fin d'avoir quelque chose qui marche (du moins en partie !).

Outre les compétences de BD et Texhnoweb, nous avons pu appliquer directement ce que l'on apprenait en GL. Ce qui était très interressant.

CVS est un outil formidable et, nous espérons que le département permettra de l'utiliser l'année prochaine pour les projets et les TT.

Chapitre 6

Suivi, visualisation du $\triangle$ - TT4

6.1 Points clés de notre démarche

6.1.1 Compréhension du Sujet

L'objectif de notre groupe était de réaliser toutes les fonctionnalités nécessaires au suivi d'un exercice et d'un élève. C'est une interface dont l'accés est limité aux professeurs et utilisant les fonctionnalités de presque tous les groupes (suivi, lecteur et création de la carte).

Notre travail s'est bien déroulé avec le TT3 qui codait exclusivement nos fonctions côté BD sur le suivi puisqu'ils ont répondu à nos attentes et su être disponibles quand nous avions besoin de résoudre des problèmes ensembles.

6.1.2 L'organisation du Travail

Après avoir clairement défini le travail à effectuer, nous avons établi des spécifications. Cette partie s'est bien déroulée, il n'y a eu au final que très peu de modifications par rapport à ce que nous avions prévu au niveau fonctionnalités. De plus le travail effectué côté BD était cohérent (en tout cas jusqu'à présent) et l'intégration à ce niveau s'est très bien faîte.

Pour le développement effectué sur un serveur Linux Debian avec PHP 4.2.1 et un serveur Apache, nous travaillons avec tous les groupes sur un serveur CVS.

Cet outil très pratique nous a permis de développer très indépendamment entre groupes et de vérifier perpetuellement notre intégration au sein du projet final grâce à une interface Web appropriée. Nous pouvions utiliser

des requêtes fictives pendant que tout le monde développait ses pages PHP puis certains ont commencé à renvoyer des valeurs prises dans la Base de Données et tout cela nous est apparu transparent.

Pour ce qui est de l'organisation au sein du TT, le responsable de TT (Adrien) allait seul aux réunions afin de faire un compte rendu aux autres et de discuter ensuite en groupe des spécifications. Ainsi le design des maquettes a été choisi en commun et la première réalisation est passée par les trois membres du TT avant d'être finalisée.

Pour le codage le responsable de TT s'occupait de la mageure partie des pages PHP tandis que les autres membres faisaient le lien entre les classes et les requetes BD associées. Dès que la charge de travail devenait trop lourde pour une personne (en l'occurence celui qui codait les pages PHP) nous redistribuions des parties de code a chacun. Ainsi la dernière page qui affiche les résultats du suivi a d'abord été réalisée par Adriana avant d'être intégrée au reste.
Une fois que la première maquette a été visualisée par Nicolas Malandain notre responsable de projet nous avons entièrement développé avec CVS. Ainsi nous avons pu préciser quelque peu certaines de nos spécifications et le TT3 a pu modifier ses fonctions BD afin de correspondre tout à fait à nos attentes.

6.1.3 La maquette

Nos pages sont au nombre de 4 et s'organisent comme suit :

- La première page intitulée choix_exercice_etudiant.php permet de choisir soit un Exercice, soit un Etudiant dont on veut faire le suivi.
- A partir de là on arrive respectivement sur la page choix_etudiant.php ou choix_exercice.php qui affiche, d'après le choix en première page une liste correspondante d'Etudiants ayant fait l'Exercice choisi ou bien une liste d'Exercices faits par l'Etudiant choisi.
- Ce couple de données (Etudiant, Exercice) correspond forcément à un ou plusieurs Parcours. C'est à dire que la dernière page affiche_suivi.php donne le choix entre les différents Parcours correspondants : soit à des dates différentes de Parcours, soit à pour des versions différentes d'un même Exercice.

Ce choix fait en accord avec le groupe BD d'afficher tous les Exercices indépendamment de la version s'explique par le fait qu'il serait compliqué pour l'utilisateur de faire un suivi s'il était obligé de le faire sur une version particulière. Au contraire avec un système de menu déroulant sur la dernière page (ComboBox) il a accès à plusieurs informations rapidement.

Il est à noter que pour cette dernière page nous avons eu l'autorisation d'utiliser un code Javascript très basique qui évite d'avoir à l'écran les statistiques d'un Parcours ne correspondant pas à ce qui serait selectionné dans le menu. En effet la validation d'un chagement de parcours de fait par un bouton Submit qui, s'il n'était pas pressé entrainerait une sorte de confusion.
Par défaut le parcours le plus récent est affiché à l'écran, ce qui correspondra le plus souvent à la dernière version de l'exercice et donc à celle que voulait afficher le professeur.

6.2 Problèmes rencontrés et Solutions

6.2.1 Compréhension du Sujet

Le projet de l'ENGREF étant assez compliqué à comprendre dans sa globalité tous ne sont pas partis à la même vitesse. Nous n'avons pas été les plus rapides à mettre nos pages sur CVS et c'est à ce moment là que nous nous sommes rendu compte que ce que nous avions déjà codé marchait, certes, mais n'était pas forcément en accord avec les consignes de codage. En tout cas nous avons dû repasser derrière une bonne partie de notre code afin qu'il soit conforme à ce qu'avaient fait les autres groupes et aussi pour le rendre plus propre car, après avoir repris un nouveau départ, nous avons tout de suite suivi les spécifications correctement et codé beaucoup plus intelligemment.

6.2.2 Les spécifications et la réalisation

Les difficultés commenceaient, car nos spécifications n'étaient pas forcément optimisées. Ainsi nous avions besoin d'obtenir la liste des Exercices par la fonction obtenirListeExercices() et ensuite nous récupérions pour chaque élément le nom associé et la dernière version, afin d'afficher ces informations dans le menu de choix.
Cette approche obligeait à faire 2 accès à la base pour chaque Exercice, ce qui était très contre-productif. Nous avons donc revu certaines spécificatiosn avec le TT3 pour ne créer qu'une fonction qui ne faisait qu'un seul accès base et nosu renvoyait toutes les informations d'un seul coup. Nous avons donc appris à optimiser nos requêtes et pu en supprimer certaines.

Comme pour la plupart d'entre nous c'était un premier projet de cette importance et surtout avec une interface Web, il y a eu quelques mésentendus au niveau de qui fait quoi dans les classes etc... Par exemple le groupe TT3

qui fait nos fonctions de BD avait commencé par renvoyer directement les résultats des requêtes PHP que nous devions ensuite traiter dans nos pages ! Cette erreur bien évidente pour nous qui faisions la partie Technoweb ne leur était pas apparue puisqu'ils n'avaient pas réalisé le travail a effectuer derrière. Heureusement ils étaient tout à fait disponibles et ce genre de problème a toujours été vite résolu.

6.2.3 Travail en groupe

Une erreur peut-être plus intéressante car c'est ce que nous risquons de rencontrer plus tard dans notre vie d'ingénieur sont les problèmes de communication. Par exemple un certain TT qui développait en local ses pages ne les updatait qu'hebdomadairement ce qui occasionnait souvent des problèmes de cohérence car de nombreux détails avaient changé entre temps. Heureusement je pense que les spécifications faîtes au départ étaient bonnes et elles ont toujours servi de référence quand il y avait conflit entre les différents travaux des groupes.

A ce niveau il n'était pas toujours facile de discuter car certains groupes avaient besoin de modifier les spécifications et d'autres ne voulaient absolument pas. Nous avons toujours travaillé exclusivement avec les classes Suivi, Graphe, Exercice et Utilisateur et nous avons dû réutiliser les classes et fonctions crées par les autres groupes. Il a donc fallu parfois s'adapter quand quelqu'un modifiait une classe, une fonction etc... Mais globalement ce travail purement de débuggage a été fait et les fonctions que nosu utilisons aussi bien que celles que nous avons crées sont utilisées par différents groupes.

Pour parler de la finalisation du projet il est à noter que le TT2 ayant une partie trop importante a dû arrêter son développement, ce qui a considérablement ralentit notre travail puique nous n'avons pas pu tester d'exercice réellement créé dans la base. Il a fallu demander à un membre de ce même TT2 de créer l'exercice à la main avec une interface Web PhpPgAdmin. Ce travail rébarbatif et assez complexe dû à toutes les dépendances dans la Base de données nous a retardé dans nos tests puisqu'il n'a été réalisé qu'une semaine avant l'échéance du projet !
Ce n'est que tout récemment que nous avons pu accéder à des données réelles sur un exercice.

De la même manière la carte faite en Flash par le TT5 n'a pas été testée pendant tout le projet puisqu'elle n'a jamais pu accéder à un exercice complet de la base. Cette carte que nous devions afficher fut un peu compliquée à mettre en oeuvre car nous faisions tous les tests dans le vide.

En effet le travail de ce TT s'est surtout accès sur la réalisation en Flash tandis que les spécifications n'ont pas été suffisamment etudiées et que certaines fonctionnalités n'ont été codées par personne... Il a donc fallu attendre les derniers jours du projet pour enfin voir la carte mais elle est encore en cours de développement, ce qui nous empêche pour nos propres pages d'afficher de que nosu aurions souhaité.

6.2.4 Récapitulatif

Au final nos pages marchent semble-t'il entièrement et gèrent les erreurs au niveau de la selection des Etudiants et des Exercices. Cela implique une Interface Homme Machine qui explique par exemple au professeur qu'il a choisi un Exercice qu'aucun Etudiant n'a fait. Pour des raisons de test et puisque le projet n'est pas fini les requêtes du TT3 côté BD affichent encore les exercices non finalisés dans la liste des exercices à traiter. Cela disparaitra dans la version finale du projet.

Chapitre 7

Carte - TT5

7.1 Travail à réaliser

Cette partie a pour but premier la conception et l'implémentation d'une carte dynamique de positionnement permettant à l'utilisateur, professeur ou élève de mieux visualiser l'organisation de l'exercice. Cette carte pourrait dans un deuxième temps servir à l'observation du parcours d'un élève en le comparant au parcours type défini par le professeur.

7.2 Cahier des charges

A partir d'un objet "exercice" nous devons afficher une carte regroupant les étapes, les séquences, les documents et les liens de type suivant entre les documents. Les séquences doivent être réactives (c'est-à-dire en cliquant sur une séquence on peut choisir le documents à visualiser parmi une liste). La carte doit pouvoir s'autogénérer.

7.2.1 Note

La version ne contient pas les icônes des documents appartenant aux étapes pour des questions de lisibilité. Effectivement la qualité des icônes proposées ne permet pas un affichage décent. Dans cette optique nous avons préféré ne pas les inclure plutôt que de rendre la carte illisible. Cela n'empêche en aucun lieu l'utilisateur d'accéder aux différents documents formant une séquence, un simple clic sur la séquence en question et l'utilisateur voit la liste des documents.

7.2.2 Exemple de carte dynamique de positionnement

La carte dynamique est une synthèse de ces deux cartes, il suffit d'ajouter les liens visibles sur la premiere figure sur le schéma de l'exercice.

FIG. 7.1 – Exemple de Carte Dynamique de Positionnement avec les étapes

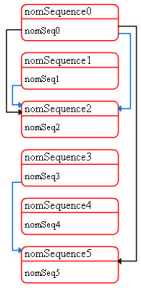

FIG. 7.2 – Exemple de Carte Dynamique de Positionnement avec aperçu des liens - les liens visités apparaissent en bleu tandis que les liens vierges sont en noir

7.3 Choix d'implémentation

Novices dans cette approche du Web graphique, nous avons cherché les différentes possibilités permettant de décrire facilement dans un langage informatique la création d'une carte. Ceci étant avec la contrainte de l'interfaçage avec du PHP objet. Après moultes recherches sur Internet, flash semblait être la seule solution envisageable dans notre cas.

Hormis le fait que flash semble être l'unique solution disponible pour répondre à ce cahier des charges, la réactivité et la portabilité de l'Action Script en font un premier choix. Or flash est une marque déposée de Macromedia et n'est donc pas un logiciel libre. Ainsi ce choix n'est pas recevable dans notre étude.

Quelques efforts supplémentaires nous ont conduit vers Ming. Cette bibliothèque, disponible sur `http://ming.sourceforge.netest` en tout point similaire à la solution propriétaire évoquée précédemment. Ming est une librairie de fonctions en C pour la génération d'objets au format SWF (Flash) intégrant un interfaçage avec d'autres langages de programmation tel C++, PHP, Python et Ruby. Nous avons donc préféré la bibliothèque Ming à Flash. Ainsi nous avons les avantages d'un code libre et la réactivité du flash.

7.4 Principe fondamental

La carte s'affiche en cascade. Chaque élément appelle les éléments qui dépendent de lui, ainsi on obtient le schéma présent sur la page suivante. Ce choix permet une création dynamique de la carte. Effectivement, le script de génération du SWF parcourt la structure d'un graphe de manière descendante afin de dessiner en cascade les différents éléments de la carte.

7.5 Détail (très peu détaillé) des fonctions

Chaque classe possède une méthode genererSWF qui lui permet de se dessiner.

7.5.1 Classe Graphe

`genererSWFGraphe($_largeur, $_hauteur,$_movie_name)`

Cette fonction permet de créer la carte pour un graphe donné et elle l'enregistre dans $_movie_name.swf (il ne faut pas préciser .swf dans $_movie_name). Les trois paramètres permettent de définir la hauteur, la largeur et le nom

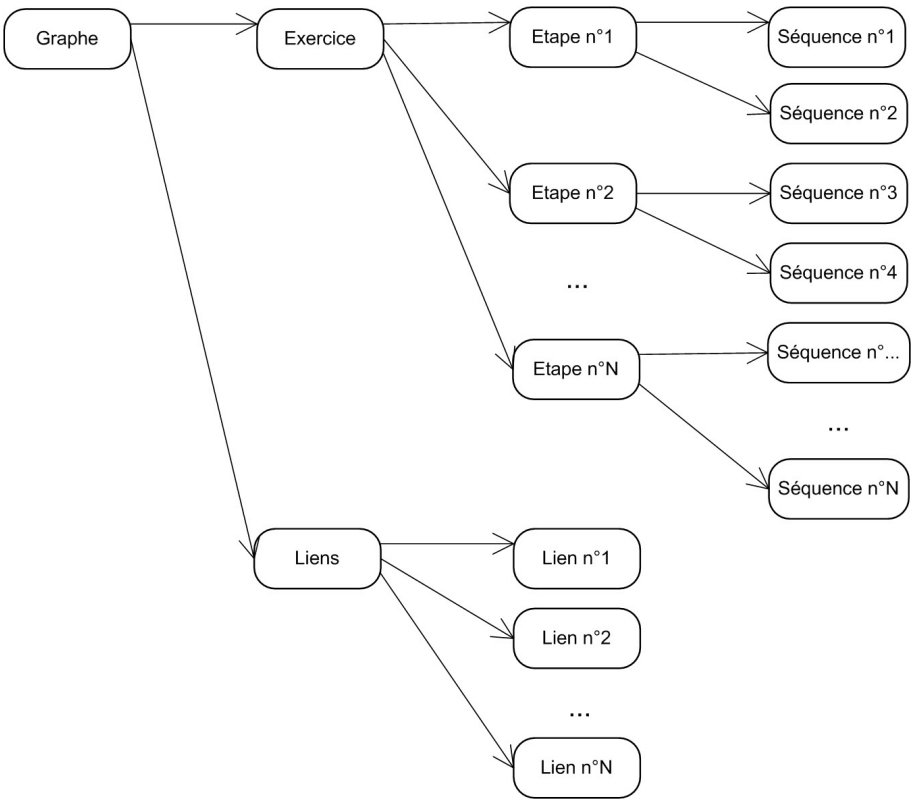

FIG. 7.3 – Une architecture en cascade

du clip. La fonction va généré un clip flash et instancier un exercice, pour appeler la fonction genererSWFExercice dessus.

7.5.2 Classe Exercice

`genererSWFExercice(&$movie)`

Cette fonction permet de générer tout la partie de la carte correspondant à un exercice. La fonction reçoit en référence le clip créé par la méthode genererSWFGraphe. Ainsi toutes les éléments seront ajoutés dans ce fichier. Lors de l'instanciation de l'objet exercice, ce dernier a créé une arborescence d'étapes qui ont elles mêmes récupérées les informations significatives des séquences leur appartenant. La fonction genererSWFExercice appelle donc la fonction genererSWFEtape sur chaque instance d'étape le constituant.

7.5.3 Classe Etape

```
genererSWFEtape(&$m, $x1, &$y1)
```

Il s'agit ici de générer "les boites étape", c'est-à-dire le cadre autour de n séquences de l'étape, ainsi que le libélé de l'étape. La fonction reçoit le clip flash ainsi que les coordonnées à partir desquels l'étape peut se construire. Chaque étape "possède" un nombre de séquences sur lequel à son tour elle va appeler la fonction genererSWFSequence.

7.5.4 Classe Sequence

```
genererSWFSequence(&$m, $r, $g, $b, $x1, $x2, $y1, $y2, $y3)
```

Les paramètres sont un peu plus complexes dans ce cas. En effet,on introduit tout d'abord le clip flash. Puis on passe les valeurs des couleurs (rouge,vert, bleu) de l'étape. Viennent ensuite les coordonnées du point supérieur gauche et du coin inférieur droit. Pour finir la troisième variable est un booléen précisant si on introduit une sépartion entre le titre de l'étape et son contenu.

7.5.5 Classe Lien

```
genererSWFLien(&$m)
```

La fonction genererSWFGraphe appelle directement la fonction genererSWFLien. Cette fonction permet en connaissant les objets liens de type suivant de dessiner un trait orienté de la séquence du document de départ vers la séquence du document d'arrivée. De manière désormais classique, cette fonction reçoit en référence le clip flash de la carte.

Ici on vient créer un cadre par séquence en incluant le nom de la séquence et la couleur liée à l'étape.

7.6 Inclusion d'une carte de positionnement dans une page

Pour créer une carte il nous faut connaître le graphe qu'elle représente. Ainsi pour commencer on crée un objet graphe sur lequel on appelle la fonction genererSWFGraphe. Puis comme exposé dans les deux parties précédentes la fonction genererSWFGraphe va appeler au niveau de chaque objet une fonction genererSWF ainsi nous obtenons une carte.

Attention : la fonction genererGraphe nécessite la création d'un objet exercice et son initialisation, en fonction des attributs du graphe en question.

```
$graphe_type = new Graphe();
$graphe_type->initialiserGraphe($id_graphe_type);
//definition des parametres
$largeur = 300; $hauteur = 400; $name="movie";
$graphe_type->genererSWFGraphe($largeur, $hauteur,$name); ?>
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/
flash/swflash.cab#version=5,0,0,0" width="<? echo $largeur; ?>"
height="<? echo $hauteur; ?>">
<param name=movie value="<? echo $name.".swf"; ?>">
<param name=quality value=high>
<embed src="<? echo $name.".swf"; ?>" quality=high
pluginspage="http://www.macromedia.com/shockwave/download/
index.cgi?P1_Prod_Version=ShockwaveFlash"
type="application/x-shockwave-flash" width="<? echo $largeur;?>"
height="<? echo $hauteur; ?>">
</embed>
</object>
```

7.7 Problèmes rencontrés

7.7.1 Le premier problème

A l'origine nous penchions pour une création de carte à la volée, et envoyer le flot de données directement dans le `mbd`. Malheureusement cette solution n'est pas adéquate. La meilleure solution disponible est de préciser avant l'appel de la fonction genererSWFGraphe le nom du fichier dans lequel sera enregistrée la carte. Pour ensuite indiquer au embed l'adresse de ce fichier.

7.7.2 Mais où placer les liens ?

Les liens pour beaucoup sont des passerelles entres documents et figurent donc au plus profond de l'arbre représentant la structure de l'exercice. Or cela pose de gros problèmes pour l'affichage des liens. Effectivement les liens sont des éléments d'une structure de graphe. Par conséquent, ils sont de niveau supérieur aux documents et aux séquences (puisque nous souhaitons afficher les liens entre les séquences).

Ainsi nous avons eu recours à la création d'éléments liens au niveau de la classe graphe et ce parallèlement à la création de l'exercice.

7.7.3 Un autre problème de taille : A qui le devoir de créer un initialiseur pour Exercice ?

Pour dessiner un graphe nous faisons appel à la fonction genererSWFExercice afin de récupérer toute la structure d'un exercice. Cette étape nécessite donc l'utilisation d'une fonction d'initialisation d'exercice. Or il n'existe aucune fonction de ce type dans la classe Exercice, seule une fonction initialiserExercice4Lecture est présente. Nous avons donc repris cette fonction. Néanmoins nous ne pouvons que déplorer l'inexistence d'une fonction correcte d'initialisation d'un graphe et de l'exercice. Nous estimons que cette démarche était du ressort d'un autre groupe.

Le sujet étant la création d'une carte de positionnement, nous nous sommes basés sur la définition des classes en Techno Web. Ainsi l'ensemble des fonctionnalités pour la création d'un graphe est disponible en supposant qu'un objet exercice puisse être instancié de façon correcte afin de satisfaire les exigences inhérentes à la carte.

7.8 Améliorations envisageables

Pour l'instant les deux types de cartes, c'est-à-dire la carte de positionnement à proprement parler et la carte de description de parcours, sont cliquables. Une option devrait permettre le choix des liens actifs ou non. Les liens, bien qu'ayant toutes les qualités nécessaires ne sont pas reconnus comme visités ou non. Néanmoins la mise en application devrait etre plus que possible puisque il existe un attribut visité au sein de la classe lien.

En outre l'utilisateur a la possibilité de cliquer sur toutes les séquences de l'exercice. Or lors du parcours d'un exercice il ne devrait pas etre possible de cliquer sur la séquence en cours d'exploration. Ceci est un autre point à modifier.

L'affichage des icones dans les séquences permet d'avoir une meilleure vue d'ensemble de l'exercice et notamment des documents qui le composent. Nous ne les avons pas intégrés pour des raisons de lisibilité, étant donné le style des icones et leur qualité la lecture n'aurait pas été facilitée. Mais ceci reste possible il suffirait d'ajouter une fonction genererSWFDocument qui afficherait l'icone correspondant au type du document.

D'autre part les noms des séquences sont trop grand et dépasse donc de la zone qui leur est allouée dans la carte. Il serait de bon ton de calculer

la longueur de cette chaine et de faire un retour à la ligne si nécessaire. A première vue ceci semble simple à réaliser, mais ce changement influe sur le calcul des liens, qui lui est relativement complexe. En conclusion ce serait un interessant sujet d'étude.

7.9 Conclusion

Les fonctions générant la carte sont opérationnelles. On dispose des deux affichages possibles :

- la carte de position avec tous les liens de type suivant,
- le parcours d'un étudiant ou le parcours type.

Il est à noter que toutes les options possibles ne sont pas encore disponibles (cf. améliorations possibles). Seule ombre au tableau : nous déplorons l'absence d'un style défini spécifiquement (notamment une cohérence dans les couleurs entre la carte et le style général de la page).

Chapitre 8

Conclusion

8.1 Résultat final

L'application n'est pas fonctionnelle. Dans la mesure où nous avons bien spécifié notre travail, les bases de l'application doivent être bonnes. Cela permettra au étudiants qui achèveront son développement de conserver le travail déjà effectué.

8.1.1 Ce qui est fonctionnel

- L'authentification des utilisateurs
- Les outils d'aide à l'analyse des parcours (Delta)
- L'accès au contenu du site (Consultation)
- La carte de positionnement (à 90%)

8.1.2 Ce qui reste à faire

- Un gros travail d'intégration de tous les composants déjà développés
- La création d'exercice à terminer

8.1.3 Perspectives

Il semblerait que le projet soit prolongé par une UV libre au semestre prochain. L'objectif du groupe qui va prendre ce travail en charge sera finaliser l'application et d'en délivrer une version pleinement opérationnelle à l'ENGREF.

Chapitre 9

Annexes

Dans la suite sont joints tous les comptes-rendus rédigés à la suite des réunions. Ils contiennent l'essentiel des discussions et problèmes que nous avons eu, et les décisions prises. Ils sont le reflet de notre travail au cours de ces 7 semaines, et des moyens que nous avons mis en œuvre.

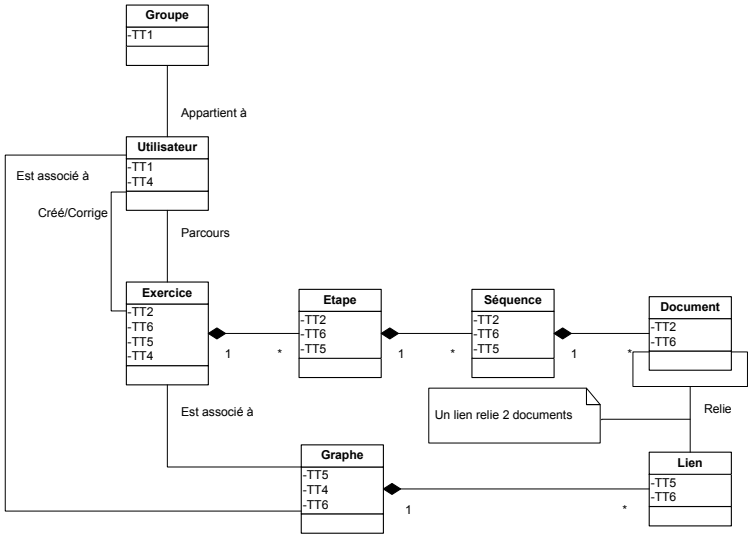

FIG. 9.1 – Diagramme de classes de l'ensemble du projet