ResSocUTT

Création d'un réseau social

Rapport de projet

Réalisé par :
Florent LUCET

Encadré par :
Marc LEMERCIER

Au sein de l'UV :
LO07, Programmation web

Lors du semestre de :
Printemps 2013

SOMMAIRE

INTRODUCTION

Le projet ResSocUTT consiste à entreprendre entièrement la conception et la réalisation d'un réseau social numérique pour les étudiants de l'UTT. Il s'agit de définir des utilisateurs –représentés ici sous la forme de profils– ainsi que leurs caractéristiques et les liens (relations) présents entre eux. De plus, l'administrateur du site disposerait d'un outil de surveillance et d'analyse du réseau social afin de détecter des comportements inappropriés.

Le projet ResSocUTT se base bien entendu sur nos connaissances web, mais pas seulement ! Pour mener à bien le projet, il nous est également nécessaire de bien comprendre les besoins du client, et d'avoir un minimum de notions d'interface web et de base de données. Loin de se limiter à la réalisation, la programme web nécessite également une phase d'analyse des besoins du client et de conception. Une bonne conception assurera une bonne réalisation. Dans cette optique, ce rapport comportera alors au moins une partie pour chacune de ces étapes du processus de développement du projet.

I) Analyse des besoins

Le premier point que l'on peut remarquer, c'est que la persistance des données est au centre du projet. En effet, le but est ici de créer un réseau social. Pourrait-on vraiment appeler réseau social un site qui ne conserverait pas les données de ses utilisateurs les semaines passant ? En plus de la non-praticité évidente que risquerait un tel système, il est certain que celui-ci ne remplirait pas son but premier : permettre aux utilisateurs d'avoir des informations sur les autres utilisateurs et de pouvoir visionner leurs photographies. Un réseau social sans persistance des données, ce n'est pas un réseau social ! D'où la nécessité absolue d'une persistance des données intégrée au site. La persistance des données sera permise grâce à une base de données, bien plus maintenable et pratique à utiliser que d'autres possibilités de persistances des données tels qu'un tableau Excel ou l'utilisation du CSV.

Un second point qui saute également aux yeux est la primordialité de l'existence d'éléments permettant à chacun de modifier ses propres informations. Imaginons un réseau social où tout le monde peut modifier les données de tout le monde. Certains s'amuseraient à modifier les données d'autres utilisateurs et le site serait clairement ingérable. Ce besoin est primordial avant tout parce qu'il permet d'instaurer un équilibre sur le site en bloquant l'accès à une modification de données qui ne nous concernent pas directement. Cela peut être représenté de différentes manières, et nous nous servirons ici de la plus utilisée : les comptes utilisateurs. En plus d'être une méthode relativement facile à mettre à place et pratique à utiliser, elle permet une distinction bien nette entre plusieurs groupes d'utilisateurs (administrateurs, membres, etc.).

Un troisième point, également relativement important, est la possibilité de contrôler la visibilité de nos données. Ce besoin permet aux utilisateurs une véritable souplesse d'utilisation en ne pouvant décider de montrer leurs données qu'aux personnes voulues. Un réseau social sans contrôle de la visibilité de nos données, c'est un réseau social flexible et moins capable de s'adapter à l'utilisateur. C'est suffisamment important pour y prêter une réelle attention.

Un quatrième point, plus général et systématique chez tous les types de sites web, consiste à avoir un site web à la fois pratique et rapide à utiliser. Le site web mal conçu et peu agréable à utiliser car pensé trop rapidement, on en a tous fait l'expérience un jour. Quelques clics en trop pour une action pourtant simple, des validations à n'en plus finir ou des éléments importants suffisamment mal placés pour qu'on ne les remarque pas, et c'est un lot d'utilisateurs, dégoûté, qui va préférer aller voir ailleurs ! Dans ces conditions, autant bien penser à la conception avant d'attaquer la réalisation, non ?

II) Conception : partie base de données

II.1) Données essentielles

Pour construire une base de données, il est nécessaire, dans un premier temps, de réfléchir aux éléments qu'on souhaite y insérer. Pour cela, une petite relecture des informations données s'impose, dans l'intention d'y relever les tables, attributs et cardinalités qui nous permettront de construire notre base de données. Une fois celle-ci faite, des éléments-clés apparaissent comme de très probables futures tables :

- Une table **Compte** permettrait le stockage de l'identifiant de l'utilisateur, ainsi que le stockage de son mot de passe, de son prénom, de son nom, de son sexe, de son programme, de son semestre, de la visibilité de son programme, de celle de son semestre, de ses compétences et enfin de ses photographies.

- Des tables **Type_competence**, **Type_relation**, **Type_visibilite** et **Type_evenement** permettraient de gérer avec souplesse les types de compétences (« Merise », « PHP », « LaTeX », etc.), types de relations (« Je connais… », « Je travaille avec… », etc.), types de visibilités (« Public », « Amis », « Privé ») et types d'évènements (« Inscription », « Ajout ou modification de paramètres », etc.).

- Des tables **Competence**, **Relation** et **Evenement** pour faire le lien avec leurs homologues version « Type » décrits précédemment. La table **Competence** contiendrait alors des couples idCompte-idTypeCompetence. La table **Relation**, un peu particulière, contiendrait des triples idCompteSource-idCompteDestinataire-idTypeRelation. Et enfin, la table **Evenement** contiendrait un duo idCompte-idTypeEvenement avec en plus la description de l'évènement et la date de l'évènement. L'absence d'une table **Visibilite** serait due au fait que les idVisibilite sont stockés dans la table **Compte**.

- Une table **Photo** contenant les chemins où seront stockées les photos sur le serveur.

II.2) Modèle Conceptuel des Données

Une fois les données essentielles repérées, un Modèle Conceptuel des Données fût réalisé. Pour le concevoir, le logiciel Win'Design fût utilisé.

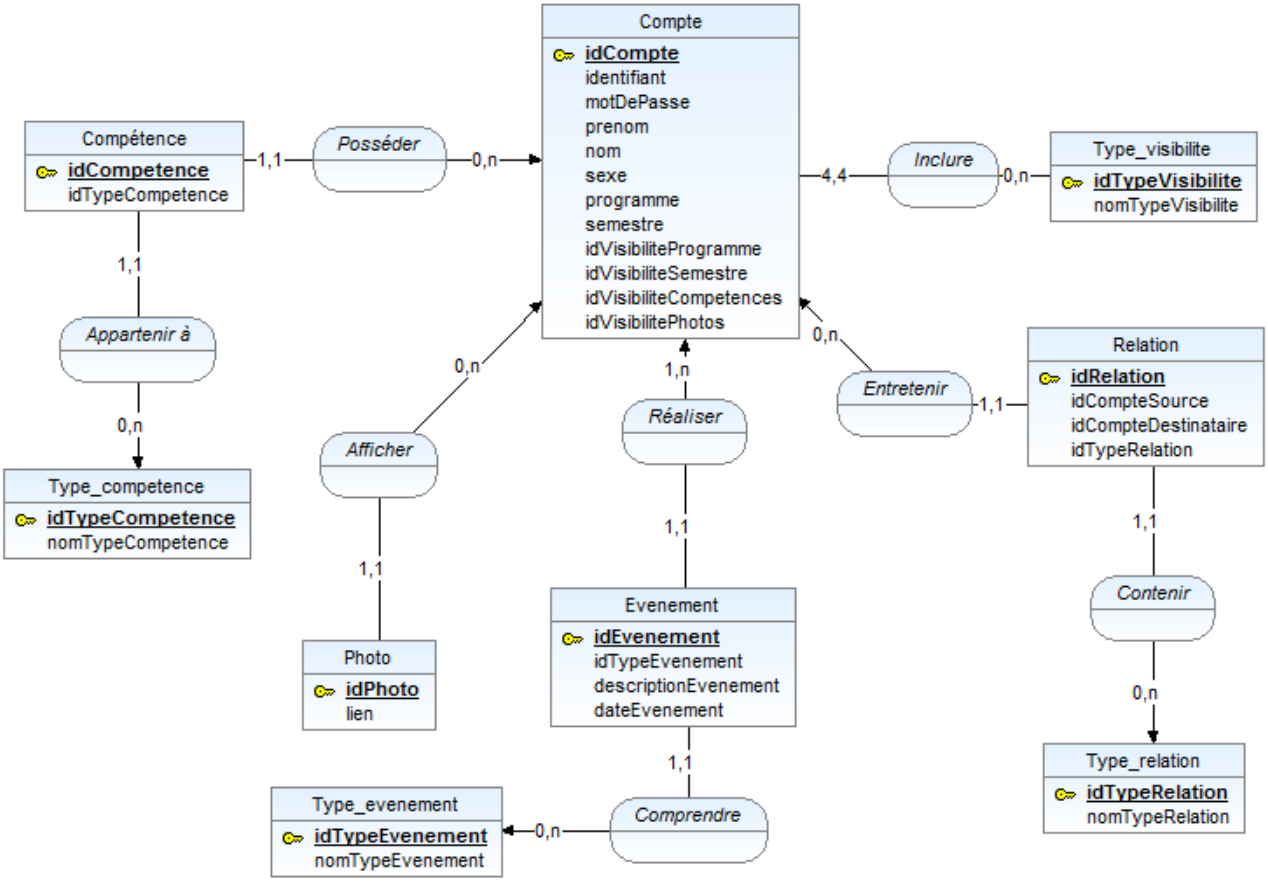

Diagramme 1 – Modèle Conceptuel des Données

Ce diagramme comporte quelques cardinalités intéressantes par plusieurs abords. Déjà, les cardinalités en « 1, 1 » deviendront des clés externes dans le Modèle Logique des Données. Cela concerne donc les tables **Competence**, **Evenement**, **Relation** et **Photo**. Un compte réalise de 1 à n évènements et non de 0 à n pour une bonne raison : l'inscription compte comme un événement. Il est assez logique que tous les tables de types comportent « 0, n » de leur côté. En effet, pour un type d'évènement (par exemple), il existe de 0 à n évènements correspondants. Dans ce diagramme, une cardinalité apparaît comme très particulière : « 4, 4 » du côté de la table **Compte**, entre la table **Compte** et la table **Type_visibilite**. Cette cardinalité est due au fait qu'il y a quatre « idVisibilite » pour un compte, et que les quatre seront forcément remplis, et donc, associés à quatre types de visibilités correspondants.

II.3) Modèle Logique des Données

Le Modèle Conceptuel des Données réalisé, il était temps de concevoir le Modèle Logique des Données. Grâce à Win'Design, il fût généré à partir du Modèle Conceptuel des Données. Certaines cardinalités deviennent alors des clés externes.

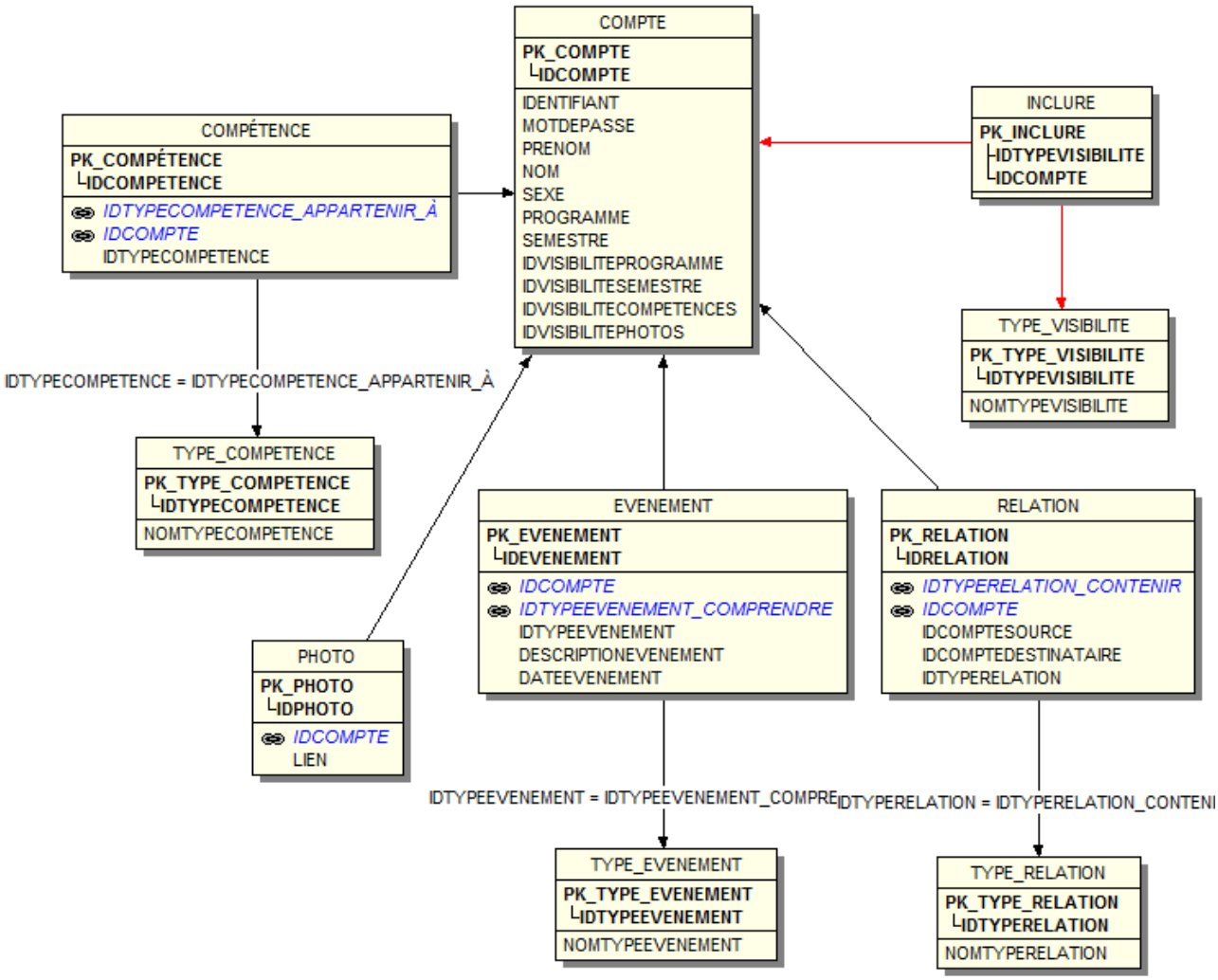

Diagramme 2 – Modèle Logique des Données

On y voit l'apparition des clés externes « idTypeCompetence_Appartenir_À » et « idCompte » dans la table **Compétence**. En effet, une compétence associe un compte à un type de compétence. Il est alors normal de retrouver des clés externes « idCompte » et « idTypeEvenement_Comprendre » pour la table **Evenement**, où les comptes sont associés à des types d'évènements ; et des clés externes « idTypeRelation_Contenir » et « idCompte » pour la table **Relation**, où les comptes sont associés à des types de relations. La table **Inclure**, quant à elle, associe le compte à quatre types de visibilités, grâce aux quatre « idVisibilite » de la table **Compte**. Enfin, une photographie n'étant associée qu'à un seul compte, il est logique de retrouver une clé externe « idCompte » dans la table **Photo**.

II.4) Modèle Physique des Données

Le Modèle Physique des Données fait suite au Modèle Logique des Données. Comme pour ce dernier, il fût généré par Win'Design. Cependant, la version obtenue en dump via phpMyAdmin étant légèrement plus complète, elle fût préférée au fichier généré par Win'Design.

```
-- phpMyAdmin SQL Dump
-- version 3.5.1
-- http://www.phpmyadmin.net
--
-- Client: localhost
-- Généré le: Mer 26 Juin 2013 à 00:33
-- Version du serveur: 5.5.24-log
-- Version de PHP: 5.4.3

SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;



--
-- Base de données: `ressocutt`
--

-- --------------------------------------------------------

--
-- Structure de la table `competence`
--

CREATE TABLE IF NOT EXISTS `competence` (
  `idCompetence` int(10) NOT NULL AUTO_INCREMENT COMMENT 'Identifiant de la
compétence',
  `idCompte` int(10) NOT NULL,
  `idTypeCompetence` int(10) NOT NULL COMMENT 'Nom de la compétence',
  PRIMARY KEY (`idCompetence`),
  KEY `idCompte` (`idCompte`),
  KEY `idTypeCompetence` (`idTypeCompetence`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=0 ;

-- --------------------------------------------------------
```

```
--
-- Structure de la table `compte`
--

CREATE TABLE IF NOT EXISTS `compte` (
  `idCompte` int(10) NOT NULL AUTO_INCREMENT COMMENT 'ID du compte',
  `identifiant` char(30) NOT NULL COMMENT 'Identifiant utilisé pour le compte',
  `motDePasse` char(30) NOT NULL COMMENT 'Mot de passe utilisé pour le compte',
  `prenom` char(30) NOT NULL COMMENT 'Prénom de l''utilisateur du compte',
  `nom` char(30) NOT NULL COMMENT 'Nom de l''utilisateur du compte',
  `sexe` char(10) NOT NULL COMMENT 'Sexe de l''utilisateur du compte',
  `programme` char(10) NOT NULL COMMENT 'Programme de l''utilisateur du compte',
  `semestre` int(10) NOT NULL COMMENT 'Semestre de l''utilisateur du compte',
  `idVisibiliteProgramme` int(10) NOT NULL COMMENT 'Identifiant de visibilité du
programme du compte',
  `idVisibiliteSemestre` int(10) NOT NULL COMMENT 'Identifiant de visibilité du
semestre du compte',
  `idVisibiliteCompetences` int(10) NOT NULL COMMENT 'Identifiant de visibilité
des compétences du compte',
  `idVisibilitePhotos` int(10) NOT NULL COMMENT 'Identifiant de visibilité des
photos du compte',
  PRIMARY KEY (`idCompte`),
  UNIQUE KEY `login` (`identifiant`),
  KEY `idCompte` (`idCompte`),
  KEY `idCompte_2` (`idCompte`),
  KEY `idVisibiliteProgramme` (`idVisibiliteProgramme`),
  KEY `     idVisibiliteSemestre` (`idVisibiliteSemestre`),
  KEY `idVisibiliteCompetences` (`idVisibiliteCompetences`),
  KEY `idVisibilitePhotos` (`idVisibilitePhotos`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=0 ;

-- --------------------------------------------------------

--
-- Structure de la table `evenement`
--

CREATE TABLE IF NOT EXISTS `evenement` (
  `idEvenement` int(10) NOT NULL AUTO_INCREMENT COMMENT 'Identifiant de
l''évènement',
  `idCompte` int(10) NOT NULL COMMENT 'Identifiant du compte ayant effectué
l''évènement',
  `idTypeEvenement` int(10) NOT NULL COMMENT 'Description de l''évènement',
  `descriptionEvenement` char(120) NOT NULL COMMENT 'Identifiant du type
d''évènement',
  `dateEvenement` datetime NOT NULL COMMENT 'Date de l''évènement',
  PRIMARY KEY (`idEvenement`),
  KEY `idCompte` (`idCompte`),
  KEY `idTypeEvenement` (`idTypeEvenement`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=0 ;

-- --------------------------------------------------------
```

```
--
-- Structure de la table `photo`
--

CREATE TABLE IF NOT EXISTS `photo` (
  `idPhoto` int(10) NOT NULL AUTO_INCREMENT COMMENT 'Identifiant de la photo',
  `idCompte` int(10) NOT NULL,
  `lien` char(150) NOT NULL COMMENT 'Lien de la photo sur le serveur du site
Web',
  PRIMARY KEY (`idPhoto`),
  UNIQUE KEY `lien` (`lien`),
  KEY `idCompte` (`idCompte`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=0 ;

-- --------------------------------------------------------

--
-- Structure de la table `relation`
--

CREATE TABLE IF NOT EXISTS `relation` (
  `idRelation` int(10) NOT NULL AUTO_INCREMENT COMMENT 'Identifiant de la
relation',
  `idCompteSource` int(10) NOT NULL COMMENT 'Identifiant du compte source de la
relation',
  `idCompteDestinataire` int(10) NOT NULL COMMENT 'Identifiant du compte
destinataire de la relation',
  `idTypeRelation` int(10) NOT NULL COMMENT 'Identifiant du type de la
relation',
  PRIMARY KEY (`idRelation`),
  KEY `idCompteSource` (`idCompteSource`),
  KEY `idCompteDestinataire` (`idCompteDestinataire`),
  KEY `idTypeRelation` (`idTypeRelation`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=0 ;

-- --------------------------------------------------------

--
-- Structure de la table `type_competence`
--

CREATE TABLE IF NOT EXISTS `type_competence` (
  `idTypeCompetence` int(10) NOT NULL COMMENT 'Identifiant du type de compétence
du compte',
  `nomTypeCompetence` char(30) NOT NULL COMMENT 'Nom du type de compétence du
compte',
  PRIMARY KEY (`idTypeCompetence`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Structure de la table `type_evenement`
--

CREATE TABLE IF NOT EXISTS `type_evenement` (
  `idTypeEvenement` int(10) NOT NULL,
  `nomTypeEvenement` char(50) NOT NULL,
  PRIMARY KEY (`idTypeEvenement`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
-- --------------------------------------------------------
```

```
--
-- Structure de la table `type_relation`
--

CREATE TABLE IF NOT EXISTS `type_relation` (
  `idTypeRelation` int(10) NOT NULL COMMENT 'Identifiant du type de relation',
  `nomTypeRelation` char(30) NOT NULL COMMENT 'Nom du type de relation',
  PRIMARY KEY (`idTypeRelation`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Structure de la table `type_visibilite`
--

CREATE TABLE IF NOT EXISTS `type_visibilite` (
  `idTypeVisibilite` int(10) NOT NULL COMMENT 'Identifiant du type de
visibilité',
  `nomTypeVisibilite` char(20) NOT NULL COMMENT 'Nom du type de visibilité',
  PRIMARY KEY (`idTypeVisibilite`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;



--
-- Contraintes pour les tables exportées
--


--
-- Contraintes pour la table `competence`
--

ALTER TABLE `competence`
  ADD CONSTRAINT `competence_ibfk_2` FOREIGN KEY (`idTypeCompetence`) REFERENCES
`type_competence` (`idTypeCompetence`),
  ADD CONSTRAINT
 `competence_ibfk_1` FOREIGN KEY (`idCompte`) REFERENCES `compte` (`idCompte`);


--
-- Contraintes pour la table `compte`
--

ALTER TABLE `compte`
  ADD CONSTRAINT `compte_ibfk_5` FOREIGN KEY (`idVisibiliteSemestre`) REFERENCES
`type_visibilite` (`idTypeVisibilite`),
  ADD CONSTRAINT `compte_ibfk_1` FOREIGN KEY (`idVisibiliteProgramme`)
REFERENCES `type_visibilite` (`idTypeVisibilite`),
  ADD CONSTRAINT `compte_ibfk_3` FOREIGN KEY (`idVisibiliteCompetences`)
REFERENCES `type_visibilite` (`idTypeVisibilite`),
  ADD CONSTRAINT `compte_ibfk_4` FOREIGN KEY (`idVisibilitePhotos`) REFERENCES
`type_visibilite` (`idTypeVisibilite`);
```

```
--
-- Contraintes pour la table `evenement`
--

ALTER TABLE `evenement`
  ADD CONSTRAINT `evenement_ibfk_4` FOREIGN KEY (`idTypeEvenement`) REFERENCES
`type_evenement` (`idTypeEvenement`),
  ADD CONSTRAINT `evenement_ibfk_3` FOREIGN KEY (`idCompte`) REFERENCES `compte`
(`idCompte`);


--
-- Contraintes pour la table `photo`
--

ALTER TABLE `photo`
  ADD CONSTRAINT `photo_ibfk_1` FOREIGN KEY (`idCompte`) REFERENCES `compte`
(`idCompte`);


--
-- Contraintes pour la table `relation`
--

ALTER TABLE `relation`
  ADD CONSTRAINT `relation_ibfk_3` FOREIGN KEY (`idTypeRelation`) REFERENCES
`type_relation` (`idTypeRelation`),
  ADD CONSTRAINT `relation_ibfk_1` FOREIGN KEY (`idCompteSource`) REFERENCES
`compte` (`idCompte`),
  ADD CONSTRAINT `relation_ibfk_2` FOREIGN KEY (`idCompteDestinataire`)
REFERENCES `compte` (`idCompte`);



/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

III) Conception : partie aspects web

III.1) Navigation : enchaînement des fenêtres

La navigation, sur un site web, implique souvent la présence de liens hypertextes et de changements de fenêtres. Cela donne donc lieu à un enchaînement de fenêtres différentes, qui nous mèneront vers de nouvelles fonctionnalités du site web. Dans le cas du réseau social ResSocUTT, il était primordial d'y penser le plus tôt possible. C'est pourquoi, une fois la base de données terminée et avant de commencer la réalisation du site Web, il a été décidé de prévoir la navigation à travers le site, l'enchaînement entre ses différentes fenêtres. À ce but, il a été réalisé un Schéma d'Enchaînement des Fenêtres simplifié, grâce au logiciel Edraw Max.

Légende du Schéma d'Enchaînement des Fenêtres simplifié :

- **Couleurs des cercles :**
  - **Cercles verts :** pages en rapport avec l'accueil.
  - **Cercles rouges :** pages d'erreur.
  - **Cercles bleus :** pages du profil accessibles par tous les membres.
  - **Cercles gris :** pages du profil uniquement accessibles par les administrateurs.
  - **Cercles oranges :** pages en rapport avec la recherche de membres.
  - **Cercles roses :** pages en rapport avec le profil d'un autre membre.
- **Liens entre les cercles :**
  - **Flèches :** liens unidirectionnels.
  - **Traits :** liens bidirectionnels.
- **Symboles des cercles :**
  - **Maison :** pages en rapport avec l'accueil.
  - **Drapeau rouge :** pages d'erreur.
  - **Un cadenas :** pages réservés à tous les membres.
  - **Deux cadenas :** pages uniquement réservés aux administrateurs.

Toutes les pages intermédiaires de confirmation ne sont pas présentes sur le Schéma d'Enchaînement des Fenêtres simplifié. En effet, le schéma, voulu le plus exhaustif possible, commençait déjà à être particulièrement chargé ; et rajouter les nombreuses pages de confirmation n'aurait eu pour résultat que l'alourdir davantage, alors que l'essentiel ne se trouvait pas dans les pages de confirmation. Les pages d'erreur redirigent l'utilisateur vers la page d'accueil s'il n'est pas connecté, vers la page d'affichage de son profil s'il l'est.

Comme nous pouvons le voir sur le schéma, la connexion d'un membre lui donne une très grande liberté. Ainsi, il peut, sur quasiment toutes les pages, accéder aux quatre principales pages : l'affichage du profil, la modification du profil, la galerie d'images et la recherche d'un ou plusieurs profil(s). À ces possibilités se rajoute, sur certaines pages, la possibilité d'aller directement sur le profil d'un autre membre (par exemple, s'il fait partie de nos relations) ; ainsi que la possibilité d'accéder aux pages de la table de traçage et des informations de traçage, si on est administrateur.

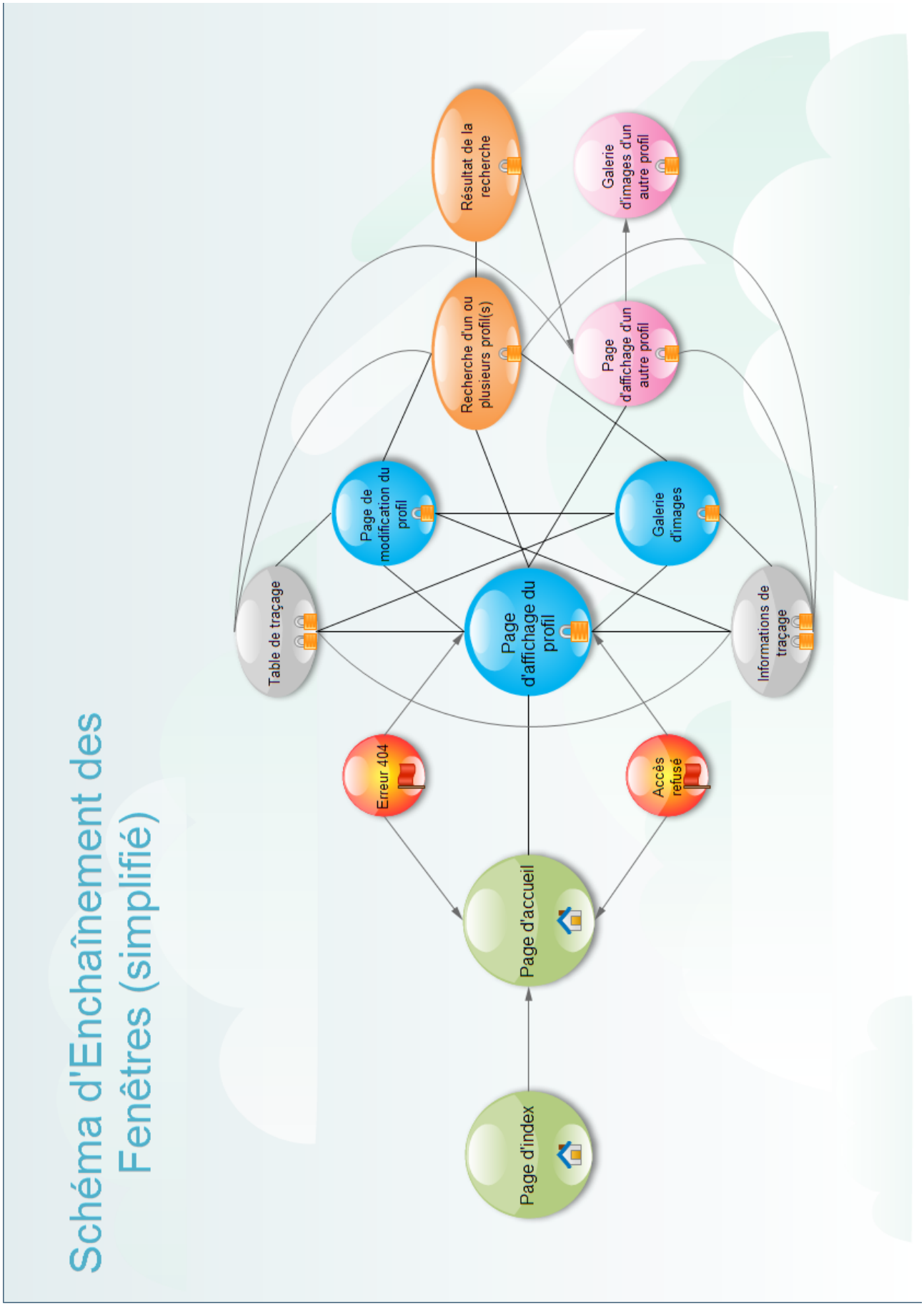

Schéma 1 – Schéma d'Enchaînement des Fenêtres (simplifié)

III.2) Interactions : positionnement et fonctionnalités des éléments

Les interactions sur le site web sont assez nombreuses. C'est pourquoi cette partie sera décomposée page par page.

La page d'index indique simplement de cliquer sur un hyperlien pour rentrer sur le site. La page d'accueil comporte beaucoup plus d'éléments. Déjà, la partie gauche de la page, qui souhaite bienvenue à l'utilisateur et lui propose de s'inscrire si cela n'est pas déjà fait. L'utilisateur peut alors s'inscrire en remplissant les différents champs à droite de l'écran, puis valider. Attention, tous les champs doivent être remplis avant validation ; sinon, l'inscription n'aboutira pas et le site web demandera à l'utilisateur de remplir les champs vides. Une fois tous l'inscription terminée, l'utilisateur arrive sur la page d'affichage de son profil. Revenons à la page d'accueil. Celle-ci comporte également un pied-de-page, présent sur toutes les pages du site (à la différente des pages de confirmation). La page d'accueil comporte également une entête, avec une certaine particularité. En effet, l'entête de l'accueil, contrairement à l'entête des autres pages du site, comporte deux champs (identifiant et mot de passe) destinés à l'utilisateur déjà inscrit qui souhaiterait se connecter. Celui-ci n'a alors qu'à rentrer son identifiant et son mot de passe, puis valider. Si l'identifiant ne correspond pas au mot de passe, un message s'affichera et l'indiquera à l'utilisateur. Celui-ci, pour se connecter, doit donc rentrer une paire identifiant-mot de passe valable, c'est-à-dire déjà enregistrée dans la base de données, c'est-à-dire un utilisateur déjà inscrit.

La page d'affichage du profil se compose comme toutes les pages d'un membre connecté : l'entête se situe en haut, le pied-de-page en bas, le menu de navigation entre les différentes pages dans une petite partie sur toute la hauteur entre entête et pied-de-page, à gauche, et l'importante parte consacrée au contenu de la page sur toute la hauteur entre entête et pied-de-page, à droite. À partir de cette base commune, chaque page affiche un contenu différent.

La page d'affichage du profil affichera les différentes caractéristiques du profil, à la seule exception des photos. La page de modification du profil permettra de modifier les différentes caractéristiques du profil, à la seule exception, encore une fois, des photos. La galerie d'images permet d'ajouter des photos. La recherche permet de trouver d'autres membres du site en les recherchant grâce à leurs différentes caractéristiques. L'affichage du profil d'un autre membre permet de sélectionner puis valider nos relations avec ce membre. On peut alors retourner sur la page de ce membre via notre page d'affichage du profil, étant donné que celui-ci affiche également nos relations avec d'autres membres. Tout ce qui implique un changement dans la base de données basé sur le choix d'un membre utilise un formulaire.

Deux autres pages sont également disponibles dans le menu de navigation, si le membre connecté est administrateur. La première page, la table de traçage, permet d'afficher un tableau de tous les évènements s'étant déroulés pour les divers membres, depuis la création du site. La seconde page, les informations de traçage, permet de connaître divers informations particulières, tels que le nombre d'évènements réalisés par tous les membres (tableau classé en ordre décroissant de nombre d'évènements), la réputation de tous les membres (tableau classé en ordre décroissant de réputation) et la liste de tous les binômes de travail.

III.3) Visuel : charte graphique

Pour la charte graphique, des nuances de bleu ont été choisies. Le but étant de donner une atmosphère aérienne au site. Ainsi, il n'est pas étonnant de voir que le fond est une photographie légèrement nuancé du ciel. Cela correspond bien à un UTT à la fois tourné vers le futur et ouvert vers le ciel (intérieur de l'ovale), sachant que le bleu est une couleur très lié à la technologie, l'informatique, le high-tech. On pense par exemple au bleu métallisé, ou aux nombreux sites de technologie et d'information qui adopte, pour couleur dominante, le bleu. C'est aussi une couleur qui donne une impression de liberté. La liberté dans le ciel, mais aussi la liberté dans l'eau. C'est une également une couleur du repos, une couleur douce, agréable, absolument pas violente (ce que pourrait être le rouge, par exemple). Quand des personnes vont sur un réseau social, ne cherchent-elles pas un peu à se reposer ?…

Toujours dans cette optique de repos et de liberté, les cadres ont, pour la plupart, des coins arrondis. Les confirmations et redirections se font via un petit cadre situé au centre, montrant encore plus l'immensité du fond-ciel, et donc, donnant davantage une impression de liberté. Les boutons poussoirs gris tranchant particulièrement avec cette atmosphère, il m'a fallu chercher des boutons correspondant davantage à la charte graphique du site. C'est pourquoi les boutons poussoirs finaux, bien plus grand que les originaux, sont si différents des petits boutons poussoirs gris. Les couleurs des polices varient, mais ne sont jamais très différentes du noir, du bleu à teinte plus ou moins foncée, ou du blanc.

IV) Réalisation : problèmes rencontrés et solutions apportées

Des problèmes de plusieurs ordres ont été rencontrés aux cours de ce projet :

- **Authentification :**
  - **Comment faire pour faire persister la connexion d'un membre ?**
    - La solution trouvée fût d'utiliser des variables de session. Utiliser des champs « hidden » ou des cookies sans particularité (donc pas des variables de session) étaient également des solutions possibles, mais elles comportaient des inconvénients que les variables de session n'avaient pas. De plus, les variables de session sont plus pratiques à utiliser et sont justement faites pour la persistance de la connexion d'un membre. Autrement dit, pourquoi vouloir réinventer la roue, quand ce qu'on a nous suffit déjà très bien ?…
  - **Comment faire pour empêcher un utilisateur non-connecté d'aller sur des pages uniquement accessibles aux membres ?**
    - Le fait d'utiliser des variables de session facilitait vraiment la tâche ici. En effet, il suffit de vérifier, en début de page réservée, si la variable de session correspondant à l'identifiant (donc à la connexion) est vide. Si elle l'est, c'est que l'utilisateur entre sur une zone qui ne lui est pas autorisée. Il est alors redirigé vers la page d'accès refusé, qui elle-même ramène l'utilisateur à la page d'accueil.
  - **Comment faire pour empêcher un membre d'aller sur des pages uniquement accessibles aux administrateurs ?**
    - Ce problème ajoute une nouvelle dimension au problème précédent. C'est le même type de problème, mais légèrement différent. Ici de ne pas afficher les hyperliens vers les pages administrateurs si l'identifiant n'est pas l'un des administrateurs. Cependant, si on se limite à cela, l'utilisateur pourra rentrer dans la zone administrateur en saisissant le lien absolu de la zone administrateur. C'est là où l'utilisation d'un fichier .htaccess et d'un fichier .htpasswd prend place. Le .htaccess permet de protéger le répertoire contenant les pages administrateur en demandant un identifiant et un mot de passe. L'identifiant et le mot de passe sont, quant à eux, inscrits à l'intérieur du .htpasswd. Ainsi, même si un membre trouvait le lien absolu de la zone administrateur, il aurait encore affaire à un mot de passe, et ne pourrait pas aller plus loin. Une erreur le redirigera sur la page d'accès refusé, et il sera ramené à la page d'affichage du profil s'il est connecté ; sinon, à la page d'accueil.

```
AuthName "Page administrateur"
AuthType Basic
AuthUserFile "C:\Program Files (x86)\wamp\www\ResSocUTT\administration\.htpasswd"
Require valid-user
```

Capture d'écran 1 – Fichier .htaccess

```
admin:admin
```

Capture d'écran 2 – Fichier .htpasswd

- **Photographies :**

  - **Comment faire persister les photographies ?**
    - Il a été décidé de stocker le chemin de l'image sur le serveur et uniquement le chemin de l'image, plutôt que de la stocker dans son intégralité dans la base de données (grâce au type Blob). Cela est préférable à l'autre solution, car si on stockait les images dans la base de données, ses performances risqueraient d'en pâtir et elle serait fortement alourdi par le stockage d'images. Ainsi, une photographie transversée est placée dans un document propre au numéro d'identifiant du compte de l'utilisateur, et seul le chemin de l'image est conservé dans la base de données.

  - **Comment faire pour que deux photographies ne portent jamais le même nom ?**
    - Les photographies sont renommées avant leur stockage sur le serveur et le stockage de leur chemin. Pour les renommer, on utiliser l'algorithme md5 sur la fonction uniqid() (génère un identifiant unique, préfixé, base sur la date et heure courante en microsecondes). Autrement dit, le nom est totalement aléatoire et il y a extrêmement peu de chances que deux photographies tombent sur le même nom.

- **Requêtes :**

  - **Comment faire une requête de recherche qui s'adapte aux choix de l'utilisateur ?**
    - On utilise une chaîne de caractères de base, comportant la partie invariante de la requête, c'est-à-dire le SELECT. En fonction des choix de l'utilisateur, on ajoute à cette même chaîne telle ou telle restriction. La chaîne de caractères finale, relativement longue et complètement fonctionnelle, permet l'affichage du tableau contenant les membres correspondant à nos restrictions.

  - **Quelle est l'une des requêtes possibles pour trouver les réputations des différents membres (partie administration) ?**
    - La solution retenue, après un certain temps de réflexion, fût directement l'une des bonnes solutions. J'ai choisi d'utiliser des requêtes imbriquées au sein du premier SELECT, ce qui donne une requête assez longue, mais fonctionnelle. Voici cette requête : « SELECT identifiant, ((SELECT COUNT(idRelation) FROM relation AS R1 WHERE R1.idCompteDestinataire = C.idCompte) / ((SELECT COUNT(idRelation) FROM relation AS R1 WHERE R1.idCompteDestinataire = C.idCompte) + (SELECT COUNT(idRelation) FROM relation AS R2 WHERE R2.idCompteSource = C.idCompte))) AS reputation FROM compte AS C ORDER BY reputation DESC; ».

- **Divers :**

  - **Comment faire pour que l'utilisateur soit redirigé, s'il entre sur une page non-trouvée du site ?**
    - La solution trouvée fût d'utiliser un fichier .htaccess. Celui-ci permet de faire des redirections automatiques vers des pages, dans le cas où se produisent certaines erreurs (dont la fameuse erreur 404, « page non-trouvée »). C'est donc ce qui a été utilisé en vue de rediriger vers une page d'erreur 404 qui, elle-même, redirige vers la page d'affichage du profil en cas de connexion, la page d'accueil sinon.

  - **Comment réaliser un menu en accordéon ?**
    - Ce menu a été rendu possible grâce au JavaScript, et plus particulièrement grâce à jQuery. Un peu de CSS en supplément, et cela a donné le menu en accordéon que nous pouvons voir sur la page d'affichage de notre profil, pour montrer nos relations avec d'autres membres du site.

  - **Où est ce bug ?**
    - Cette question-problème fait en réalité référence à un ensemble de problèmes. Ce sont tous ces petits problèmes qui font perdre une heure ou deux alors qu'il suffit de rajouter un guillemet ou un point-virgule à un endroit où on l'avait oublié. Par exemple, je me suis retrouvé à chercher pendant deux heures le problème dans la requête destinée à afficher les réputations des membres. J'y avais utilisé le champ « idCompteDestination » inexistant de la table Relation au lieu du champ « idCompteDestinataire », alors même qu'en dehors de cela, ma requête était bonne dès le départ. J'ai eu le même type de problème, auparavant, avec une requête où j'avais mis « idTypeEvenement » au lieu de… « idTypeEvenement ». Avec une espace à la fin. La base de données l'avait enregistré, et cette espace était proche de l'indétectable. Il fallait décider de modifier le champ pour avoir une chance de la voir, la colonne étant trop petite sur phpMyAdmin. Ces problèmes sont –de loin– ceux qui m'ont fait perdre le plus de temps, lors de ce projet. D'où l'importance de le signaler.

CONCLUSION

Au final, ce projet aura été très instructif. Déjà, il m'aura permis de réutiliser les notions vues en cours et en TP. Que ce soit l'utilisation d'un serveur Apache, de scripts PHP, de pages HTML, de feuilles de styles CSS, d'instructions Javascript, de formulaires, d'une base de données relationnelle ou même de variables de session, une grande partie des notions vues au sein de l'UV LO07 auront été réutilisées. Il aura également été mobilisé des connaissances d'autres cours, concernant plutôt les bases de données que la programmation web. Ainsi, aurais-je eu l'occasion de concevoir un MCD, un MLD et un MPD, dans l'optique de faire la base de données la plus robuste possible. De plus, l'UV NF16 n'aura pas été la seule autre UV connexe à ce projet. L'UV EG23 (Interface Homme-Machine) permettait aussi de concevoir, avec davantage de recul, l'interface du site web, ses interactions, le positionnement de ses éléments.

Ce projet a, de plus, une certaine portée. La programmation web est de plus en plus utilisée en informatique, au détriment des autres programmations. Aujourd'hui, il est beaucoup plus probable de tomber sur un stage ou un emploi en lien avec la programmation web qu'hier. La programmation web est devenue incontournable dans l'informatique d'aujourd'hui. Les bases de données également. Il y a donc de fortes chances que notre futur stage ou notre futur emploi nous demande de réutiliser des notions vues à travers l'UV LO07. D'autant que ce projet se rapproche, à certains niveaux, d'un projet professionnel : le cycle « analyse des besoins, conception, réalisation », l'utilisation de diagrammes de conception en vue de répondre aux besoins d'un client, des délais, un travail en équipe à gérer correctement et équitablement, l'utilisation de connaissances spécialisées en vue de répondre aux besoin d'un client, penser l'interface par rapport à ses utilisateurs…

Quoi qu'il en soit, je serai très certainement amené, à l'avenir, à réutiliser ces connaissances de programmation web, ainsi que les logiciels/API utilisés pour ce projet (Win'Design, Edraw Max, jQuery). D'où l'importance de continuer à travailler cette partie de l'informatique, qui me sera très probablement utile prochainement…