**AP3 2021 – Forma'Flix**

PARTIE I – Prise en main de l’existant

Tout d’abord, analysons la base de données et la schématisation conceptuelle de ce projet.

Pour commencer, nous observons une contrainte unique pour le champ *identifiantVideo* de la table **FORMATION** qui permet d’affecter une et une seule vidéo par formation.

Celle-ci est également présente dans la table **VIDEO** pour la même propriété. Cette contrainte permet d’empêcher la possession de deux identifiants identiques pour deux données différentes.

Pour continuer, l’association **DEVELOPPER** devient une table dans la base de données. Cette association permet de lier la table **FORMATION** à la table **COMPETENCE**.

Ce choix a été effectué car cela permet de trier les formations par compétence. Il sera donc possible par exemple de rassembler des formations par compétence. (dans une requête SQL via un GROUP BY par exemple)

Les paramétrages de cette table concernent les champs *idFormation* et *idCompetence*, ces derniers sont également liés à la table **COMPETENCE** et **FORMATION**.

A propos des contraintes étrangères, nous observons la notion de ON DELETE NO ACTION qui permet d’empêcher la suppression ou modification si cela impacte d'autres données dû à la présence de la clé étrangère. Cependant, sur une clé étrangère il est préférable de laisser l'attribut d'origine RESTRICT car il n'y a habituellement pas de modification effectuée sur un AUTO-INCREMENT.

Pour ce qui est de ON DELETE RESTRICT, cela signifie que l’on ne peut pas supprimer une valeur référencée par une clé étrangère. RESTRICT a le même effet que NO ACTION.

Concernant la notion de ON UPDATE CASCADE, cela signifie que lorsqu'une formation est mise à jour/supprimée, les données de cette table référencées dans d'autres tables le sont aussi.

S’il est nécessaire, nous pouvons utiliser le SET NULL qui permet de ne pas supprimer les données mais de remplacer leur valeur par NULL tout en restant présente dans la base de données. Cela peut être utile en cas d’archivage par exemple.

En ce qui concerne le PHP, les mots de passe sont cryptés de manière à ce que personne ne puisse voir les mots de passe en cas d’accès nuisible à la base.

Lors de la phase d’initialisation du projet, nous avons choisi de posséder deux comptes distincts pour pouvoir manipuler la BDD en tant qu’utilisateur (tests) et en tant qu’admin (configuration).

COTE WEB

Du coté applicatif, la gestion des mots de passe dans le code C# consiste à comparer le mot de passe en clair saisi par l’utilisateur et celui présent dans la base de données qui est crypté. Il faut donc utiliser la méthode Verify qui permet de décrypter le mot de passe hashé. La librairie utilisée est BCrypt.

Lors de la connexion, le nom de l’utilisateur connecté est récupéré et indiqué en haut de la page principale de l’application : l’information est transmise d’une vue à l’autre grâce à la classe Controleur de VmodeleC.

Lors de l’ajout d’une formation : le code de cet évènement consiste à vérifier la saisie de l’utilisateur puis d’ajouter la formation après avoir chargé les précédentes formations afin de pouvoir récupérer la clé primaire de la formation qui est en auto-increment. Dans un premier temps, on utilise la méthode *AjoutFormation* en ajoutant une formation et une compétence. Dans un second temps, on utilise la méthode *AjoutDevelopper* car il faut procéder à l’ajout de cette formation dans la table **DEVELOPPER** qui est composée de l’*idFormation* et l’*idCompetence.*