BREVET DE TECHNICIEN SUPÉRIEUR

SERVICES INFORMATIQUES AUX ORGANISATIONS

Option : Solutions logicielles et applications métiers

U6 – CYBERSÉCURITÉ DES SERVICES INFORMATIQUES

Hackat'Innov++

BTS BLANC - Novembre 2021- Lycée Bellepierre

Proposition de corrigé

Barème

Dossier A	Gestion des participants	30 points
Dossier B	Application de votes	30 points
Dossier C	Application serveur	25 points
Dossier D	Sécurisation de l'architecture	15 points
	TOTAL	100 points

Dossier A : Mme Kosmalski

Dossiers B - C - D : M. Defaÿ

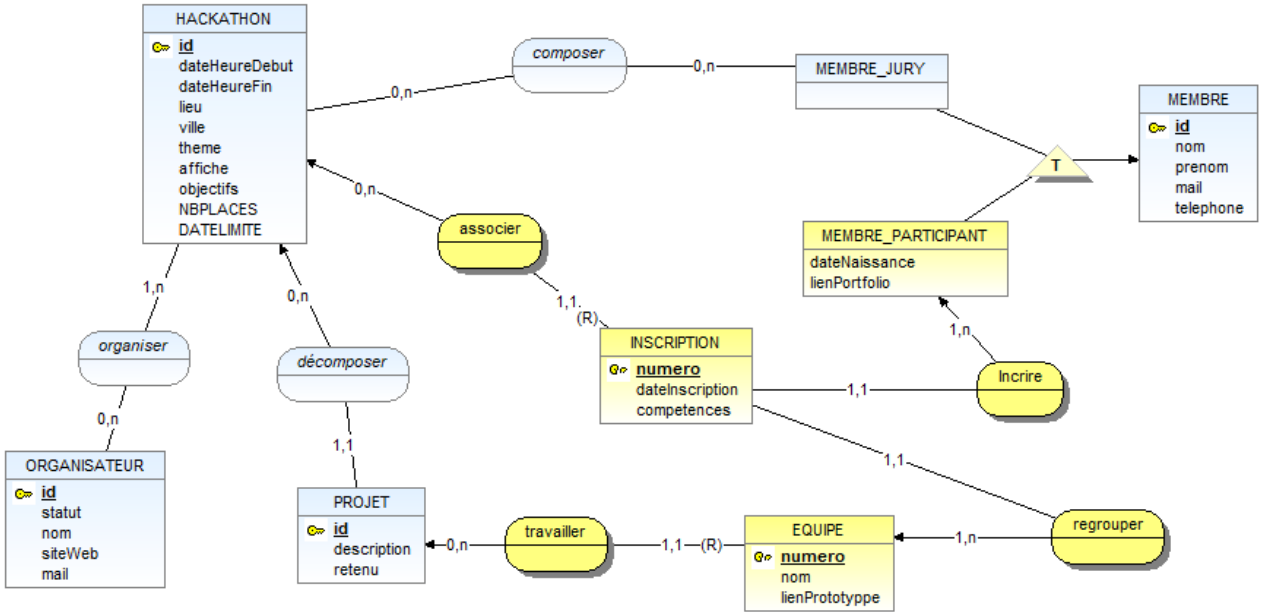
Dossier A – Gestion des participants

Mission A.1 – Évolution de la base de données pour la gestion des hackathons

Question A.1

Proposer une modification de la base de données utilisée par l'application existante *Hackat'Orga* prenant en compte les inscriptions, la gestion des équipes.

Représentation du schéma entité-association :



Bloc 1 (3,5 points) : nouvelle entité MEMBRE_PARTICIPANT + contrainte d'héritage de TOTALITE qui exprime qu'un membre participant peut être aussi un membre de jury.

Bloc 2 (4,5 points) : nouvelle entité EQUIPE + 2 associations « regrouper » et « travailler » avec identifiant relatif.

Bloc 3 (4 points) : nouvelle entité INSCRIPTION + 2 associations « inscrire » et « associer » + 2 nouveaux attributs dans l'entité HACKATON.

Mission A.2 – Sécurisation du processus d'inscription

Question A.2.1

Identifier les données personnelles récoltées lors du processus d'inscription. Préciser si parmi ces données certaines sont éventuellement des données sensibles.

(2 points)

Lors du processus d'inscription, **toutes les données collectées sont personnelles :** nom, prénom, mail, téléphone, date de naissance.

Parmi ces données personnelles collectées, **aucune ne peut être considérée comme sensible**.

Question A.2.2

a. Analyser le traitement réalisé sur ces données dans ce contexte en définissant leur finalité et leurs bases légales.

b. Préciser si les données collectées dans ce contexte auraient pu être minimisées.

(1,5 points)

a. Ces données personnelles collectées puis stockées dans la base de données le sont pour permettre à des utilisateurs de **s'inscrire à un hackaton**. Légalement, l'application doit informer l'utilisateur de cette collecte et lui demander tacitement son **consentement**.

b. Les données personnelles collectées sont déjà minimales et, dans ce contexte, n'auraient pu être minimisées.

Question A.2.3

Rédiger une courte note à l'attention de Mme Mabille en lui indiquant, si nécessaire, les étapes devant être réalisées pour une mise en conformité au RGPD.

(5 points)

En se basant sur le document 2, le candidat devra rédiger une courte note à l'attention de Mme Mabille (la présentation de la note sera à noter) avec les principaux éléments suivant :

Rédaction d'une politique de confidentialité. Ce document devra être accessible aux utilisateurs à tout moment de la procédure et devra contenir à minima les informations suivantes :

- Un consentement explicite de chaque utilisateur (case à cocher par l'utilisateur lui-même ; pas de pré-coche) ;
- L'information claire et accessible des données qui sont collectées et dans quel but et leur durée de conservation par l'organisation ;
- L'énoncé des droits de l'utilisateur sur ces données personnelles et les moyens de les exercer ;
- Une mise en place d'une politique de sécurisation de ces données.

Question A.2.4

a. Indiquer les modifications à apporter au modèle relationnel, pour prendre en compte cette demande de gestion des mots de passe.

b. Rédiger une requête SQL permettant de prendre en compte ces modifications.

(3 points)

a. Pour prendre en compte cette demande, il faudra ajouter un nouvel attribut à la table MEMBRE, nommé, `motPasse` de type chaine de caractère. Pour permettre la vérification de la durée de validité maximale de 3 mois, il est également nécessaire d'ajouter un second attribut dans la table MEMBRE, nommé `dateValid` de type Date.

b.
```
ALTER TABLE MEMBRE ADD COLUMN motPasse VARCHAR(30), dateValid DATE ;
```

Question A.2.5

Rédiger les requêtes SQL permettant d'afficher les besoins en informations ci-dessus.

(3,5 points)

```
SELECT PROJET.id, description
FROM PROJET INNER JOIN HACKATON
      ON idHackaton = HACKATON.id
WHERE DATE(dateHeureDebut) = '2022-12-03'
AND lieu = "La Défence"
AND ville = "Paris";
```

(3 points)

```
SELECT id, nom, prenom
FROM MEMBRE
WHERE id NOT IN
      (SELECT idMembreJury FROM COMPOSER);
```

Dossier B – Application de votes (30pts)

Question B.1 (14pts) : Expliquer l'erreur rencontrée pour le problème 1 et proposez :
- Une solution pour éviter que cette erreur fasse « planter » l'application
- Deux solutions différentes pour éviter que cette erreur ne se produise lors de la saisie des votes

Explication (4pts):

Ce message indique que la contrainte « note_inf_ou_egal_5 » n'a pas été respectée. On tente donc d'insérer un vote avec une note non comprise entre 0 et 5. Le contrôle côté serveur (base de données) est assuré, l'intégrité des notes est donc garantie !

Éviter le plantage (4pts):

Pour éviter l'arrêt brutal de l'application, il faudrait intercepter les exceptions (bloc « try catch ») lors de l'exécution de la requête voire vérifier la valeur de la note reçue dans le code PHP (moins bon en termes de dépendance car ne respecte pas le principe de responsabilité unique « SRP[1] »). En cas d'erreur, le script côté serveur (PHP) pourrait effectuer une redirection vers une page d'erreur ou transmettre au client (le navigateur) un indicateur (variable URL, session, cookie) lui permettant de gérer l'affichage d'un message d'erreur.

Éviter l'erreur (6pts):

Si la vérification côté serveur doit être maintenue pour garantir l'intégrité des données (note valide), la vérification doit également être faite côté client pour améliorer l'expérience utilisateur. Plusieurs solutions peuvent être envisagées :

- Un contrôle HTML sur la zone de texte :
  `<input name="txtNote" id="idNotel" type="number" min="0" max="5">`
- Un contrôle effectué en JavaScript avant envoi des données du formulaire,
- Une liste déroulante des notes possibles,
- 5 boutons radio pour les notes acceptées.

Question B.2 (8pts) : Expliquer pourquoi l'erreur rencontrée pour le problème 2 est possible et modifiez le schéma conceptuel (ou le schéma relationnel) de la base de données afin de résoudre ce problème. Ne reproduisez que les entités (ou relations) nécessaires à votre modification.

Explication (4pts):

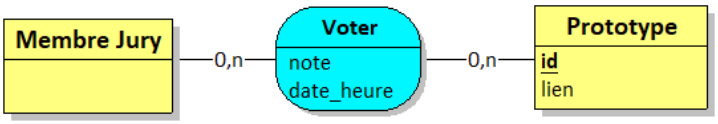
L'association ternaire « voter » se traduit par une relation dont la clé primaire sera composée de 3 champs : id (du membre jury) + dateHeure + id (du prototype). La contrainte d'unicité de la clé portant sur ces trois champs, il est donc possible d'insérer plusieurs notes pour un même juge et un même prototype.

Solution (4pts) :

[1] SRP : Single Responsibility Principle

La clé primaire ne doit concerner que les champs id (du membre de jury) + id (du prototype). Elle passe alors en troisième forme normale.

La relation devient alors : Voter(idMembreJury, idPrototype, dateHeure, note) et l'association :



Question B.3 (7pts) : En vous basant sur la sonde réseau :
- Préciser ce qui relève d'une bonne pratique en termes de sécurité
- Expliquez en quoi cette solution ne garantit pas la confidentialité des échanges et proposez une solution sans préciser comment l'implémenter.

Bonne pratique (3pts) :

On constate que le mot de passe n'est pas transmis en clair. Un hachage côté client a été fait avant d'envoyer les données, c'est une bonne chose.

Explication (3pts) :

Cette solution ne garantit cependant pas la confidentialité du mot de passe car s'il n'est pas de complexité suffisamment élevée, il pourra être retrouvé à partir de son empreinte (dictionnaires de mots de passe, *brut force*, *rainbow table*).

L'ajout de sel avant hachage est une sécurité supplémentaire mais l'analyse du code côté client permettrait de le retrouver et de générer un nouveau dictionnaire de mots de passe.

Solution (2pts) :

La solution consiste à utiliser le protocole HTTPS. Les communications entre le client et le serveur seront ainsi chiffrées de bout en bout afin de garantir la confidentialité des échanges et donc du mot de passe.

Dossier C – Application serveur (25pts)

Question C.1 (6pts) : Identifiez une faille dans les fonctions de connexion des juges, puis proposez une solution sous forme de code.

Faille (3pts) :

Les données en provenance du formulaire ne sont pas filtrées. Le traitement n'est donc pas protégé contre les injections SQL. Le danger est d'autant plus grand que la fonction `multi_query()` permettrait d'exécuter une deuxième requête injectée.

Solution (3pts):

Il faudrait utiliser une requête préparée voire échapper les caractères spéciaux.

```php
public function auth() {

    $this->connect();
    $requete = "Select * from Membre ".
               "Where id = ? And password = ? ".
               "And id in (Select id From Composer)";

    $results = $this->db->prepare($requete);
    $results->bind_param("ss", $_POST['id'], $_POST['password']);

    $results->execute();
    $result = $results->fetch_assoc();

    ...
}
```

Question C.2 (9pts) : Pour chaque faille de sécurité relevée dans le code :
- Expliquez les conséquences (risques) encourus
- Proposez un correctif

Propriétés publiques (3pts) :

Toutes les propriétés de la classe « InscriptionDAO » sont publiques. C'est contraire au concept d'encapsulation de la POO. Le risque est faible mais une modification de ces propriétés depuis un objet connexion reste possible et cela pourrait compromettre la suite des traitements.

Solution : mettre tous les attributs en *private* (ou *protected*).

Compte root (3pts):

La connexion à la base de données se fait avec le compte root. Le risque est élevé car en cas d'injection SQL ou d'erreur de programmation, la base de données voire le serveur pourrait être compromis.

Solution : appliquer le principe du moindre privilège et créer une connexion MySQL dédiée aux traitements à réaliser sur cette base.

Méthode inscription (3pts) :

Les données du formulaire ne sont pas filtrées, le code est sensible aux injections de code JavaScript (faille XSS). Le comportement de l'application peut être perturbé, il peut y avoir vol de cookie de session, redirection, affichages intempestifs, etc. Le code est également vulnérable aux injections SQL.

Solution : échapper les caractères spéciaux, vérifier la cohérence des données, utiliser une requête préparée.

Question C.3 (10pts) : Écrivez le code permettant de générer, sous forme tabulaire HTML, les données contenues dans le fichier de log '202010_log.csv'.
Vous utiliserez le langage de votre choix, le PHP étant souhaité mais pas obligatoire.

```php
$mesLignes = getLogs('202110_log.csv');

echo '<table>';
    // Génération de l'entête
echo '<tr>';
    echo '<th>day</th>';
    echo '<th>time</th>';
    echo '<th>ip</th>';
    echo '<th>usertype</th>';
echo '</tr>';

foreach ($mesLignes as $uneLigne) {
    // Génération des lignes du tableau
    echo '<tr>';
        echo '<td>' . $uneLigne[0] . '</td>';
        echo '<td>' . $uneLigne[1] . '</td>';
        echo '<td>' . $uneLigne[2] . '</td>';
        echo '<td>' . $uneLigne[3] . '</td>';
    echo '</tr>';
}

echo '</table>';
```

Répartition des points :

1 : Appel de la fonction

1 : Génération de l'entête du tableau

4 : Parcours total des lignes

2 : Génération des lignes

2 : Génération du tableau

Dossier D - Sécurisation de l'architecture (15pts)

Question D.1 (6pts) :

Préciser les intérêts de mettre le serveur de bases de données en dehors de la zone démilitarisée (*DMZ)* pour aider Mme Mabille à compléter son étude.

Une zone démilitarisée est un sous-réseau contenant les machines accessibles depuis Internet. Ce sous-réseau est isolé du réseau local et protégé par un pare-feu. Le serveur de base de données n'a pas besoin d'être accessible depuis Internet. Le placer en dehors de la DMZ pour le rendre inaccessible depuis Internet a pour principaux objectifs de :

- Sécuriser l'accès à la base de données
- Protéger les données contre les attaques venant de l'extérieur
- Participer à la préservation de l'intégrité et de la confidentialité de la base de données.

Question D.2 (9pts):

Présenter sous forme d'un tableau le ou les éléments de l'architecture applicative concerné(s) par chacun des points de vigilance relevés.

- Sécuriser le service http (https)	- **APACHE**
- Utiliser des requêtes préparées	- **API**
- Se prémunir des attaques par injection SQL	- **API (+ Android éventuellement)**
- Se connecter via un *Wi-Fi* sécurisé	- ***Android***
- Préserver l'intégrité de données (droits d'accès – défaut de configuration)	- **BDD + API**
- S'assurer de la mise à jour des librairies et des	- ***Android*****, API, Apache, BDD**

composants logiciels	