Cas VINSDEFRANCE.COM

Proposition de corrigé

Mission 1 – Base de données du site *Web*

D4.1 Conception et réalisation d'une solution applicative
- A4.1.3 Conception ou adaptation d'une base de données

Question 1.1
Écrire en langage *SQL* les requêtes permettant de corriger ces erreurs.

```
UPDATE Composer SET pourcentage = 10 WHERE idVin 5429 AND millesime = 2020 AND idCep = 136
DELETE FROM Composer WHERE idVin = 5429 AND millesime = 2020 AND idCei =
                                        (SELECT id FROM Cepage WHERE nom = 'Chardonnay')
INSERT INTO Composer(idVin, millesime, idCep, pourcentage) VALUES (5429, 2020, 108, 90)

Autre solution :

DELETE FROM Composer WHERE idVin=5429 and millesime=2020
INSERT INTO Composer(idVin, millesime, idCep, pourcentage) VALUES (5429, 2020, 108, 90)
INSERT INTO Composer(idVin, millesime, idCep, pourcentage) VALUES (5429, 2020, 136, 10)
```

Question 1.2
Écrire en langage *SQL* les requêtes permettant d'obtenir les informations souhaitées.

a. Les noms de tous les cépages utilisés pour les vins « Blanc ».

```
SELECT DISTINCT Cepage.nom
FROM Cepage
INNER JOIN Composer ON Cepage.id = Composer.idCep
INNER JOIN Vin ON Composer.idVin = Vin.id
INNER JOIN Couleur ON Vin.idCou = Couleur.id
WHERE Couleur.nom='Blanc'
```

b. les noms des cépages les plus connus, c'est-à-dire ceux qui entrent dans la composition de plus de 200 vins différents

```
SELECT nom, COUNT(DISTINCT idVin)
FROM Cepage
INNER JOIN Composer ON Cepage.id = Composer.idCep
GROUP BY nom
HAVING COUNT(DISTINCT idVin)>200
```

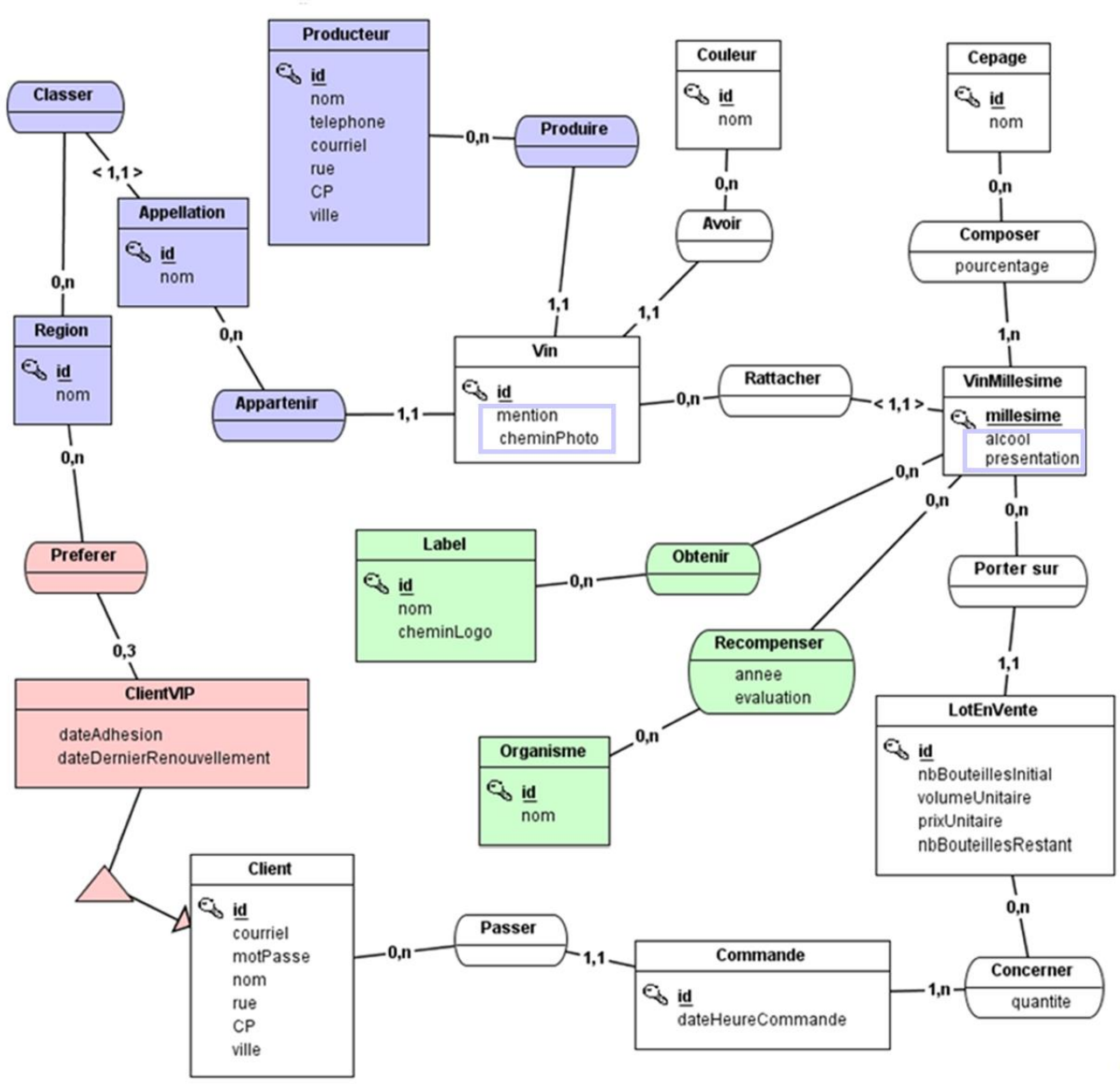
La clause DISTINCT dans le HAVING est indispensable

Le COUNT(DISTINCT idVin) dans le SELECT n'est pas attendu

Question 1.3
Proposer un schéma de structure de la base de données en complétant l'existant. Seuls les éléments du schéma existant qui sont concernés par l'évolution seront repris dans le schéma proposé

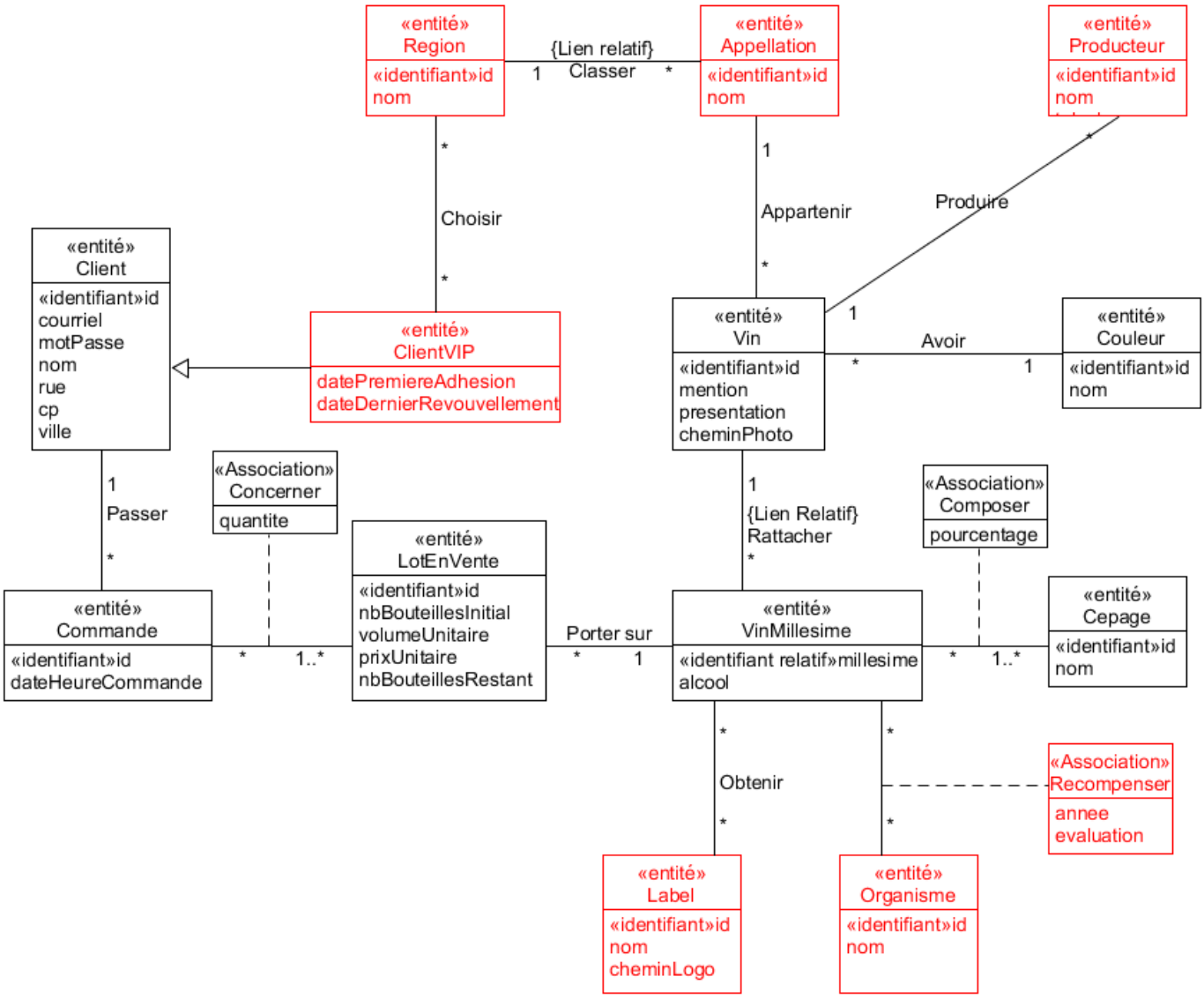
IMPORTANT : la candidate ou le candidat présentera les évolutions de la structure de la base de données en adoptant le formalisme de son choix (schéma entité-association, diagramme de classes ou encore schéma relationnel).

Schéma entité-association



* ***nbBouteillesRestant*** *est facultatif car calculable*

Diagramme de classes



* ***nbBouteillesRestant*** *est facultatif car calculable*

Schéma relationnel
(On n'a reporté ici que les nouvelles tables ou celles modifiées)

Vin (id, idCouleur**,** mention, cheminPhoto, idApp, idRegion, idProducteur)
Clé primaire : ***id***
Clés étrangères : ***idCouleur en référence à id de Couleur***
idReg, idApp en référence à idRegion, id de Appellation
idProducteur en référence à id de Producteur

VinMillesime (idVin, millesime, alcool, presentation)
Clé primaire : ***idVin, millesime***
Clé étrangère : ***idVin en référence à id de Vin***

Appellation (idRegion, id, nom)
Clé primaire : idRegion, id
Clé étrangère : idRegion en référence à id de Region

Region (id, nom)
Clé primaire : id

Producteur (id, nom, telephone, courriel, rue, cp, ville)
Clé primaire : id

Label (id, nom, cheminLogo)
Clé primaire : id

Organisme (id, nom)
Clé primaire : id

Obtenir (idVin, idLabel)
Clé primaire : idVin, idLab
Clés étrangères : idVin en référence à id de Vin
idLabel en référence à id de Label

Recompenser (idVin, idOrganisme, annee, evaluation)
Clé primaire : idVin, idOrg
Clés étrangères : idVin en référence à id de Vin
idOrganisme en référence à id de Organisme

ClientVIP (idCli, dateAdhesion, dateDernierRenouvellement)
Clé primaire : idCli
Clé étrangère : idCli en référence à id de Client

Choisir(idClientVIP, idRegion)
Clé primaire : idClientVIP, idRegion
Clés étrangères : idClientVIP en référence à idCli de ClientVIP
idRegion en référence à id de Region

On acceptera l'ajout de 3 champs : VIP booléen, dateAdhesion affecté à null et dateRenouvellement affecté à null dans la table Client au lieu de la table ClientVIP.

Question 1.4
Proposer et décrire, sans la réaliser, une solution permettant de maintenir à jour automatiquement la quantité de bouteilles disponible en stock d'un lot.

On attend la proposition d'un déclencheur (*trigger*) sur ajout/modif/suppression (*on insert* / *on update* / *on delete*) de la table concerner pour exécuter cette mise à jour directement dans la base de données.

Lors de l'ajout/modif/suppression (*insert/upodate/delete*) d'une ligne de commande dans la table concerner, il faut exécuter immédiatement une requête de mise à jour (*update*) du champ lotEnVente.nbBouteillesRestant – concerner.quantite

- **Mission 2 – Gestion des évènements**

D4.1 Conception et réalisation d'une solution applicative
- A4.1.7 Développement, utilisation ou adaptation de composants logiciels

Question 2.1
Ecrire le constructeur de la classe ClientVIP.

```
public ClientVIP(string unId, string unNom, string unPrenom, string uneAdresse, string unCourriel,
dateTime uneDateAdhesion, dateTime uneDateDernierRenouvellement) : base(unId, unNom, unPrenom,
uneAdresse, unCourriel)
{
        this.dateAdhesion = uneDateAdhesion;
        this.dateDernierRenouvellement = uneDateDernierRenouvellement;
}
```

NB : On acceptera tout type d'instruction indiquant l'appel du constructeur de la classe mère Client

Question 2.2
Ecrire la méthode *NombreTotalParticipants* de la classe Evenement.

```
public int NombreTotalParticipants()
{
   int totalParticipants = 0;

   foreach (var item in this.lesInscriptions)
       totalParticipants = totalParticipants + item.Value;

   return totalParticipants;
}
```

NB : On acceptera un parcours de la collection lesInscriptions.Values
Ou encore le remplacement de la structure itérative par l'instruction valide en *C#* :

```
        totalParticipants = this.lesInscriptions.Values.Sum();
```

Question 2.3
Ecrire la méthode *InscrireUnClientVIP* de la classe Evenement.

```
public boolean InscrireUnClientVIP(ClientVIP unClientVIP, int unNbAInscrire)
{
  boolean aRetourner = false;

  if (nbPlacesTotal - this.NombreTotalParticipants() >= unNbAInscrire &&
    this.lesInscriptions.ContainsKey(unClientVIP) == false)
   {
     this.lesInscriptions.Add(unClientVIP, unNbAInscrire);
     aRetourner = true;
   }

  return aRetourner;
}
```

NB : On ne pénalisera pas l'absence de « this »
La solution d'une boucle sans la réutilisation de la méthode NombreTotalParticipants() ne sera pas pénalisée.

Question 2.4
Ecrire la méthode *ObtenirEvenementsRentables* de la classe Catalogue.

```
public List<Evenement> ObtenirEvenementsRentables()
{
  List<Evenement> maL = new List<Evenement>();

  foreach (Evenement item in this.lesEvenements)
  {
     if (item.NombreTotalParticipants() >= item.GetNbInscriptionsMinimum())
          maL.Add(item);
  }
  return maL;
}
```

NB : La solution d'une double boucle sans la réutilisation de la méthode NombreTotalParticipants() ne sera pas pénalisée. Le candidat devra alors créer un accesseur sur le dictionnaire lesInscriptions.

Une solution basée sur une requête Linq ou une expression Lambda sera également acceptée.

Question 2.5
Compléter le code de la procédure événementielle *ConsulterEvenementsRentables_Load* de l'application *C#* qui affiche les évènements rentables du catalogue.

IMPORTANT : une méthode statique est appelée en la préfixant par le nom de la classe à laquelle elle appartient : NomDeLaClasse.NomDeLaMethode( )

```
// Déclaration d'une instance de Catalogue, destinée à contenir l'ensemble des données le concernant
// présentes dans la base de données
Catalogue leCatalogue;

// Déclaration d'une collection d'évènements destinée à contenir uniquement les événements rentables du
//  catalogue
List<Evenements> laListeEvenementsRentables;

// L'objet catalogue est chargé à partir de la base de données
// La collection des évènements rentables est chargée à partir du catalogue

        leCatalogue = CatalogueDAO.ChargerCatalogue();
        laListeEvenementsRentables = leCatalogue.ObtenirEvenementsRentables();

// La collection des évènements rentables est liée avec une grille de données afin d'afficher son contenu
//automatiquement
// La propriété DataSource de la grille de données reçoit la collection des évènements pour laquelle elle
//affiche les données
 grdEvenementsRentables.DataSource = laListeEvenementsRentables;
```

Mission 3 – Gestion des coffrets œnologiques

D4.1 Conception et réalisation d'une solution applicative
- A4.1.7 Développement, utilisation ou adaptation de composants logiciels

Question 3.1
Présenter les atouts et les contraintes de ce type d'architecture modèle vue contrôleur.

Atouts
- Séparation des entités de l'application
- Clarté de l'architecture
- Uniformisation des développements d'applications, des développements communs
- Reprise et maintenance d'applications facilitées
- Productivité accrue

Contraintes
- Complexité de l'architecture
- Appropriation de l'architecture parfois délicate

NB : Toute autre solution cohérente sera acceptée

Question 3.2
Apporter les modifications nécessaires au code *PHP* de la vue *tousLesAbonnementsVue.php* afin de l'adapter à la maquette attendue.
IMPORTANT : indiquer sur votre copie les numéros des lignes concernées.

```
1.  // La vue tousLesAbonnementsVue.php est appelée à partir de la méthode
2.  // tousLesAbonnements du contrôleur abonnementControleur.php

3.  <h1>Extrait de la liste des abonnements</h1>

4.  <?php

5.  foreach($listeAbonnements as $uneligne)
6.  {
7.    $periode = "Du ". $uneligne->dateDebut . " au " . $uneligne->dateFin;
8.    $infosClient = $uneligne->leClient()->nom . ", " . $uneligne->leClient()->adresse;

9.    $infosFormule = $uneligne->laFormule()->titre ." (" . $uneligne->laFormule()->prix . " €)";

10.   echo "<br />" . $periode . " : <strong>Client</strong> " . $infosClient ."<br />";

11.    echo " => <strong>Formule</strong> " . $infosFormule;
12.  }

13.  echo "Nombre d'abonnements : " . $nbAbonnements ;

14.  echo "<br />Nombre de clients abonnés : " . $nbClients ;
15.  echo "<br />Nombre d'abonnements moyen par abonné : " . $nbAbonnements/$nbClients;

16. ?>
```

Question 3.3
Apporter les modifications nécessaires au code *PHP* du contrôleur *abonnementControleur.php* pour permettre l'affichage de ces deux informations dans la vue *tousLesAbonnementsVue.php.*
IMPORTANT : indiquer sur votre copie les numéros des lignes concernées.

```
5.  class abonnementControleur extends Controller
6.  {
7.   // La méthode tousLesAbonnements() est appelée lors du chargement du tableau de bord de
8.   // suivi des abonnements aux formules.

9.   public function tousLesAbonnements()
10.  {

11.  // La méthode all associée au modèle abonnementModele retourne une liste de toutes les
12.  // données de la table Abonnement et des tables associées.
13.  $listeAbonnements = abonnementModele::all();

14.  // La méthode count retourne le nombre d'éléments de la liste $listeAbonnements.
15.  $nbAbonnements = $listeAbonnements->count();

16. $listeClients = clientModele::all();

17. $nbClients = $listeClients->count();

18.  // Appel de la vue tousLesAbonnementsVue.php avec la transmission des données
19.  // $listeAbonnements et $nbAbonnements.
20.  $lavue = 'tousLesAbonnementsVue';
21.  return view($lavue,['listeAbonnements'=>$listeAbonnements, 'nbAbonnements'
     =>$nbAbonnements, 'nbClients'=>$nbClients]);
22.  }
23. }
```

Mission 4 – Audit de sécurité du site *Web*

D1.2 Choix d'une solution
- A1.2.1 Élaboration et présentation d'un dossier de choix de solution technique
- A1.2.3 Évaluation des risques liés à l'utilisation d'un service

Question 4.1
Expliquer ce que la sécurisation du site web avec le protocole *HTTPS* permet de garantir aux clients de la société.

HTTPS est un protocole de communication internet qui, associé au protocole *TLS* (ou plus anciennement *SSL*), permet de garantir au client trois niveaux de protection :
- Authentification du site *Web* : le client est sûr de communiquer avec le bon site *Web*.
- Confidentialité des données : les données transmises par le client, entre le navigateur et le serveur *Web*, sont chiffrées.
- Intégrité des données : les données ne sont ni modifiées, ni corrompues durant leur transfert.

Les fonctionnalités les plus sensibles sont : la création de compte client, la connexion du client, la prise de commande et le paiement de commande par le client.

Question 4.2
Expliquer pourquoi l'utilisation de la méthode *GET* dans le formulaire *HTML* d'identification n'est pas judicieuse. Proposer une correction.

Avec la méthode *GET* les valeurs saisies par l'utilisateur sont transmises dans l'*URL*, donc le mot de passe apparait visuellement en clair à l'écran dans l'*URL*. En outre, il reste visible en clair dans l'historique du navigateur.

Il est donc préférable ici d'utiliser la méthode *POST*.

Question 4.3
a) Indiquer la requête de vérification générée à la suite de cette saisie.
b) Expliquer en quoi l'exécution de cette requête présente des risques pour *VDF* et ses clients.
c) Proposer une solution pour se protéger contre cette faille d'injection *SQL*.

a) Requête générée :

```
select * from client where courriel='' or 1=1 --' and motpasse =''
```

soit :

```
select * from client where courriel='' or 1=1
```

b) Le résultat de la requête est la liste de tous les clients.
- Pour un client, le risque est qu'un pirate puisse usurper son identité et voler ses données / passer des commandes à sa place.
- Les risques pour VDF sont la perte de confiance de la part des clients, la dégradation de son image de marque, le risque financier de devoir rembourser les clients usurpés.

c) Une solution pour s'en protéger est d'utiliser une fonction qui double les apostrophes ou une fonction qui « échappe » les apostrophes « \' » (exemple addslashes en *php*) ou utiliser une bibliothèque d'accès aux données sécurisée (exemple *PDO* en *PHP*) ou utiliser un *framework Web* (exemple *codeigniter, laravel, django*,…) qui en général prennent en charge les vulnérabilités les plus courantes. (1 proposition attendue).
Une solution côté client type vérification du format de l'adresse électronique présenterait une protection de premier niveau sans toutefois répondre complètement au besoin (cela n'empêcherait pas de forger une requête *POST* malveillante sans passer par les contrôles du formulaire)

Question 4.4
Proposer trois mesures techniques à mettre en place dans l'application pour renforcer l'authentification sur l'arrière-boutique.

- Imposer une politique de mot de passe forts : avec des expressions régulières (*regex*), avec une bibliothèque (*library*) de vérification de force de mots de passe, ...
- Forcer le renouvellement fréquent des mots de passe.
- Mettre en place une double authentification : code d'authentification par SMS sur téléphone personnel, notification par courriel sur messagerie personnelle, ...
- Utilisation de captcha pour contrer les tentatives d'attaques de force brute.
- … (3 propositions attendues)