BREVET DE TECHNICIEN SUPÉRIEUR
SERVICES INFORMATIQUES AUX ORGANISATIONS
Option : Solutions logicielles et applications métiers

U7 – CYBERSÉCURITÉ DES SERVICES INFORMATIQUES

SESSION 2025

_____

Durée : 4 heures
Coefficient : 4

_____

Matériel autorisé :
Aucun matériel ni document est autorisé.

Dès que le sujet vous est remis, assurez-vous qu'il est complet.

Le sujet comporte 20 pages, numérotées de 1/20 à 20/20.

Casterman

Ce sujet comporte 20 pages dont un dossier documentaire de 11 pages.

Barème

DOSSIER A	**Participation à l'atelier d'analyse des risques**	15 points
DOSSIER B	**Sécurisation du progiciel *BDPro***	26 points
DOSSIER C	**Gestion des droits et accès à la base de données**	23 points
DOSSIER D	**Mise en ligne de la base de données et gestion des accès**	16 points
	TOTAL	80 points

Dossier documentaire

Documents communs à tous les dossiers 10
Document commun 1 : Le processus d'édition d'une bande dessinée 10
Document commun 2 : Les fonctions des salariés de Casterman 10
Document commun 3 : Extrait de la représentation de la base de données de BDPro 11
Document associé au dossier A 12
Document A1 : Besoins de sécurité pour les récits utilisateurs (user stories) (extraits) 12
Documents associés au dossier B 13
Document B1 : Modifications et ajouts apportés à la structure de la base de données actuelle . 13
Document B2 : Déclencheur (trigger) majUtilisateur 13
Document B3 : Note fournie par le chef de projet concernant la mémorisation des fichiers au format PDF 14
Documents associés au dossier C 14
Document C1 : Diagramme des classes métier implémentées dans l'application Java (extrait) . 14
Document C2 : Classe technique ArrayList (extrait) 14
Document C3 : Memento SQL 14
Document C4 : Extrait du code Java des classes métier 15
Document C5 : Tests unitaires de la méthode possedeDroit 16
Document C6 : Méthode creerEnregLog qui écrit dans le fichier des traces (log) 17
Document C7 : Fonctions de MySQL de manipulation de dates 17
Document C8 : Structure et extrait du contenu de la table Log 17
Documents associés au dossier D 18
Document D1 : Description de la structure de l'interface de programmation API 18
Document D2 : Méthode autoriseAction de la classe AccessBdd 18
Document D3 : Classe Controle de l'interface de programmation API 18

Présentation du contexte

L'organisation cliente

Casterman est une maison d'édition de bandes dessinées et de livres pour la jeunesse créée en 1777 à Tournai (Belgique).

Le catalogue actuel s'est construit autour de *Tintin* et de *Martine*, deux œuvres plébiscitées sur toute la planète, autant par les enfants que par leurs parents. En complément de ces prestigieux piliers, Casterman publie, tant en littérature Jeunesse qu'en bandes dessinées, de grandes séries classiques (*Alix*, *Corto Maltese, etc.*) et des sagas innovantes (*Lastman*, *UW2*, *Cherub* ou *Bodyguard*).

Casterman appartient aujourd'hui au troisième groupe d'édition français et compte une soixantaine de salariés dans les bureaux de Bruxelles (Belgique) et de Paris.

Pour réaliser leur travail, les salariés de Casterman utilisent un progiciel *BDPro* qui est édité par une entreprise de services du numérique que nous appellerons ESNEdiPro.

L'entreprise prestataire de services

ESNEdiPro est éditeur de logiciels et propose également des prestations de services informatiques en développement d'applications et administration de réseaux. Elle est labellisée ExpertCyber[1].

Un contrat de prestation de services a été établi entre Casterman et ESNEdiPro. Ce contrat définit la nature des interventions d'ESNEdiPro.

Vous faites partie de l'équipe de développement d'ESNEdiPro. Sous la responsabilité de votre chef de projet, monsieur Cibedi, votre mission consiste à participer à l'évolution du progiciel *BDPro*, en intégrant différentes contraintes liées à la sécurité informatique.

Dans un contexte d'augmentation des attaques par rançongiciels (*ransomware*), Mme Siber, la directrice des systèmes d'information de Casterman a alerté Mme Gallimard, présidente directrice générale (PDG), sur les risques liés à la cyber criminalité. Elle a défini une PSSI (politique de sécurité des systèmes d'information) et demandé à la société ESNEdiPro de renforcer la sécurité de *BDPro*. Vous participerez à l'atelier d'analyse des risques et contribuerez à renforcer la sécurité du progiciel.

Dans un contexte d'ouverture du système d'information, la base de données de *BDPro* doit être mise en ligne afin d'être utilisée par les partenaires de Casterman. Vous participerez à l'évolution du progiciel en développant de nouvelles fonctionnalités tout en prenant en compte la sécurité.

Vous vous appuierez sur le dossier documentaire mis à votre disposition.

[1] Le label ExpertCyber vise à reconnaître l'expertise des experts en cybersécurité assurant des prestations d'installation, de maintenance et d'assistance en cas d'incident.

Dossier A – Participation à l'atelier d'analyse des risques

Le progiciel *BDPro* a été développé au cours des années 2010. Depuis, les risques informatiques ont changé et les menaces se sont diversifiées. De nombreuses entreprises sont victimes de tentatives d'hameçonnage et d'attaques par rançongiciels.

Dans le cadre de la PSSI, il est nécessaire de procéder à une analyse des risques de sécurité.

Mission A1 – Évaluation des risques à partir des récits utilisateurs

Vous participez à un atelier d'analyse de risque au sein de l'équipe de développement afin d'évaluer les risques sur les récits utilisateurs (*user stories*).

Question A1.1

a) Justifier le niveau du critère de confidentialité du récit utilisateur 1.

b) Justifier le niveau du critère d'intégrité du récit utilisateur 3.

c) Justifier la différence de niveau en matière de preuve entre les récits utilisateurs 2 et 3.

Mission A2 – Gestion d'un événement redouté : attaque par rançongiciel

Les attaques par rançongiciel font toujours partie des attaques les plus fréquentes.

Aussi, Mme Siber commande une analyse d'impact à ESNEdiPro et des propositions de solutions.

Question A2.1

Citer deux impacts, sur le système d'information de Casterman, d'une attaque réussie par rançongiciel.

Question A2.2

Citer trois mesures techniques à prévoir par Casterman pour se prémunir des attaques par rançongiciel.

Comme les données personnelles enregistrées dans la base de données ne sont pas sensibles, Mme Siber estime qu'une violation des données à caractère personnel présenterait un risque mineur pour les personnes concernées et vous interroge sur les obligations réglementaires à respecter en cas de violation de ces données.

Question A2.3

a) Indiquer qui doit être obligatoirement informé de la violation de données à caractère personnel.

b) Indiquer dans quel document la violation de données doit obligatoirement être consignée.

Dossier B – Sécurisation du progiciel *BDPro*

Le progiciel *BDPro* est une application de bureau développée en *Java*. Elle accède à une base de données hébergée sur un serveur de Casterman. L'accès à *BDPro* se fait à partir des postes de travail sous *Windows* avec un mécanisme d'authentification par mot de passe. L'identifiant est formé avec la première lettre du prénom et avec le nom, par exemple *vpetit*. Le mot de passe est choisi librement par l'utilisateur, la seule contrainte étant qu'il doit avoir une longueur de six caractères.

Mission B1 – Amélioration de la sécurité de l'authentification

Actuellement, les mots de passe sont stockés en clair dans la table Utilisateur. On envisage de les supprimer de la base de données afin d'enregistrer l'empreinte (*hash*) du mot de passe ou le mot de passe chiffré.

Question B1.1

a) Expliquer pourquoi les mots de passe en clair doivent être supprimés de la base de données.

b) Indiquer quelle solution (enregistrement de l'empreinte du mot de passe ou enregistrement du mot de passe chiffré) est la plus adaptée, en justifiant.

Mission B2 – Mise en place de la nouvelle sécurité d'authentification pour le mot de passe

Mme Siber a défini une nouvelle politique de sécurité pour le mot de passe, un utilisateur doit changer son mot de passe tous les six mois et ne peut pas reprendre un des cinq derniers mots de passe utilisés.

Pour cela, la base de données a été modifiée, et un déclencheur (*trigger*) nommé *majUtilisateur* a été écrit dont le traitement attendu est le suivant :

- Le déclencheur s'exécute lors de la modification d'un enregistrement de la table Utilisateur.
- Si la modification concerne le mot de passe, alors le déclencheur appelle la fonction *mdpExisteHisto*.
- Si la fonction retourne *false*, les actions suivantes doivent être réalisées :
  - enregistrement de l'ancien mot de passe dans la table HistoMotDePasse ;
  - comptage du nombre de mots de passe enregistrés pour cet utilisateur, dans la table HistoMotDePasse ;
  - si ce nombre dépasse 5, suppression de l'enregistrement du mot de passe le plus ancien ;
  - mise à jour de la table Utilisateur par mise à jour du mot de passe actuel (champ mdpActuel) avec le nouveau mot de passe ainsi que la date et l'heure à laquelle le mot de passe actuel a été enregistré (dateHeureMdpActuel) avec la date et l'heure courante.

Après l'implémentation et quelques tests, on constate que la table HistoMotDePasse reste vide.

Question B2.1

a) Expliquer pourquoi on obtient ce résultat avec le code actuel du déclencheur (*trigger*) *majUtilisateur*.

b) Corriger le déclencheur (*trigger*) pour qu'il permette l'historisation correcte des 5 derniers mots de passe. Indiquer les numéros des lignes concernées par chaque insertion ou modification du code.

Mission B3 – Sécurisation des bandes dessinées au format *PDF*

Lorsqu'une bande dessinée est terminée, Casterman doit envoyer un document au format *PDF* à l'imprimeur pour qu'il procède à l'impression de la bande dessinée au format papier.

Mais avant cette phase finale, une bande dessinée est relue et vérifiée par l'éditeur de cette bande dessinée et ses assistants tout au long de sa réalisation. Il y a donc plusieurs documents au format *PDF* pour une bande dessinée qui sont produits et enregistrés sur un serveur de fichiers.

Lorsque la bande dessinée est jugée terminée, le document final au format *PDF* sera transmis à l'imprimeur et il n'est plus possible de produire un document au format *PDF* pour cette bande dessinée. Cela répond à une exigence de qualité, afin d'être sûr que la version de la bande dessinée imprimée soit exactement celle voulue par les auteurs et par l'éditeur.

La base de données actuelle doit évoluer afin de conserver les informations des documents au format *PDF* selon les contraintes souhaitées par monsieur Cidebi et fournies dans le dossier documentaire.

Question B3.1

Proposer, à l'aide de la note fournie par le chef de projet, une modification de la base de données actuelle pour prendre en compte la gestion des documents au format *PDF*. Seuls les éléments du schéma existant qui sont concernés par la modification seront repris dans le formalisme de votre choix.

Question B3.2

Expliquer l'intérêt d'enregistrer la signature électronique du fichier *PDF* final, pour Casterman et pour l'imprimeur.

Chaque éditeur responsable d'une bande dessinée doit faire parvenir le document final au format *PDF* à l'imprimeur, un sous-traitant situé en Vendée. Le fichier est très volumineux (2 Go) et il n'est donc pas possible de l'envoyer par courriel.

Il faut trouver une autre solution de transfert adaptée à la taille du fichier et qui garantisse la confidentialité de l'échange.

Question B3.3

Proposer une solution de transfert de documents au format *PDF* sécurisée et adaptée à la taille des fichiers.

Dossier C – Gestion des droits et accès à la base de données

Le progiciel *BDPro* est une application de bureau écrite en Java qui utilise les classes décrites dans le dossier documentaire.

Chaque utilisateur de *BDPro* a une fonction précise telle qu'éditeur de bandes dessinées, assistant d'édition, gestionnaire éditorial, etc.

Des droits doivent être attribués pour chaque fonction, représentant les opérations autorisées sur des tables de la base de données. Ces droits sont enregistrés dans la base de données utilisée par l'application *BDPro*. Le tableau suivant montre les droits de deux fonctions :

Fonction	Tables concernées	Opérations autorisées
Éditeur de bandes dessinées	Categorie	select
	BandeDessinee	select, insert, update, delete
	Auteur	select, insert, update, delete
	Realise	select, insert, update, delete
Assistant d'édition	BandeDessinee	select
	Auteur	select
	Realise	select
	Categorie	select

Mission C1 – Contrôle des droits d'accès

Dans la classe Utilisateur, la méthode *possedeDroit* dont la signature est *boolean possedeDroit* (*string uneTable, string uneOperation)* retourne un booléen indiquant si l'utilisateur a le droit ou non de réaliser l'opération *uneOperation* sur la table *uneTable*.

Des tests unitaires, fournis dans le dossier documentaire, ont été écrits pour vérifier le bon fonctionnement de la méthode *possedeDroit* avec un utilisateur ayant une fonction Assistant d'édition.

Question C1.1

Expliquer les différents cas testés par les méthodes de test de la classe UtilisateurTest.

Question C1.2

Écrire le code de la méthode *possedeDroit* de la classe Utilisateur.

Mission C2 – Mémorisation des opérations dans des fichiers de traces (*log*)

Afin de détecter les malveillances sur l'application, chaque opération réalisée par un utilisateur est enregistrée dans un fichier de traces (*log*) par la méthode *creerEnregLog*, présentée dans le dossier documentaire. Cette méthode reçoit en paramètres l'utilisateur, la date de l'opération, ainsi que le droit utilisé et les ajoute dans le fichier de traces (*log*).

Les informations mémorisées dans le fichier de traces s'avèrent insuffisantes pour une réelle traçabilité car on ne connaît pas l'adresse IP de l'équipement utilisé par l'utilisateur. L'adresse IP sera passée en tant que paramètre de type chaine de caractères à la méthode *creerEnregLog*.

Question C2.1

Compléter la méthode *creerEnregLog* afin de mémoriser l'adresse IP dans le fichier de traces (*log*), en ajoutant tous les éléments nécessaires au respect des bonnes pratiques. Indiquer les numéros des lignes concernées par chaque insertion ou modification du code.

Un script *PHP*, exécuté toutes les 10 minutes par un planificateur de tâches, lit le fichier de traces et alimente une table Log créée dans une base de données *MySQL* avec les traces réalisées lors des 10 dernières minutes.

Une application exploitant la table Log, décrite dans le dossier documentaire, est en cours d'étude mais on souhaite exploiter dès à présent la table Log afin de détecter les activités malveillantes.

On vous demande de réaliser des requêtes permettant de surveiller les opérations de la table Log.

Question C2.2

Écrire la requête donnant le nombre d'opérations de type *delete* réalisées au cours des 30 dernières minutes.

La table BandeDessinee est jugée critique. Aussi, pour surveiller les opérations de type *insert* dans cette table, on souhaite avoir une requête qui fournit le nombre d'ajouts réalisés depuis ce matin 8 h pour chaque utilisateur ayant réalisé plus de 5 ajouts. Pour répondre à ce besoin, un de vos collègues a écrit la requête suivante :

```
SELECT idUtilisateur FROM Log
WHERE nomTable = "BandeDessinee" AND operation = "insert"
AND dateHeure >= concat(current_date()," 08:00:00")
```

Question C2.3

a) Expliquer pourquoi la requête ne correspond pas au besoin exprimé.

b) Corriger la requête afin qu'elle permette la surveillance souhaitée.

Dossier D – Mise en ligne de la base de données et gestion des accès

Actuellement la base de données de *BDPro*, gérée par le SGBD *MySQL*, est hébergée chez Casterman, sur un serveur situé physiquement à Bruxelles.

Le choix a été fait de gérer l'accès aux données via une interface de programmation d'application (*API*) *REST* sécurisée, écrite en langage *PHP*.

Le progiciel *BDPro* a donc dû évoluer au niveau des accès à la base de données, en utilisant une interface de programmation *API REST* sécurisée, écrite en langage *PHP* et décrite dans le dossier documentaire.

Mission D1 – Contrôle des accès dangereux à l'interface de programmation *API REST* créée pour *BDPro*

Les utilisateurs de *BDPro* ne peuvent réaliser que des demandes permises par leur fonction. Un premier contrôle est fait directement dans *BDPro*. L'interface de programmation *API RE*ST va aussi contrôler l'utilisateur à l'origine de la demande et le droit concerné.

Les requêtes *HTTP* de type *DELETE* et de type *PUT* sont considérées comme particulièrement dangereuses pour l'intégrité de la base de données.

Question D1.1

Expliquer pourquoi les requêtes *HTTP* de type *DELETE* et de type *PUT* vers l'interface de programmation *API* sont considérées comme dangereuses pour l'intégrité de la base de données.

Dans la classe AccessBdd utilisée par l'interface de programmation *API REST*, une méthode *autoriseAction()* a été écrite pour vérifier si l'auteur de la demande a bien les droits pour consommer l'interface de programmation *API REST*. Cette méthode est présentée dans le dossier documentaire.

Vous avez la charge de sécuriser l'interface de programmation *API REST* afin que seuls les demandeurs autorisés puissent la consommer.

Dans un premier temps, seule la méthode *delete* de l'interface de programmation *API REST* sera modifiée pour prendre en compte la sécurité attendue. Les autres méthodes seront ensuite adaptées selon ce modèle.

Question D1.2

Modifier la méthode *delete* de la classe Controle afin de prendre en compte le niveau de sécurité attendu. Le code de statut *HTTP* de la requête doit être adapté en conséquence.

Mission D2 – Sécurisation des commandes des libraires

Actuellement, les commerciaux de Casterman se déplacent régulièrement chez les libraires pour présenter le catalogue des bandes dessinées. Les libraires doivent alors choisir les bandes dessinées qu'ils souhaitent vendre en boutique ainsi que la quantité désirée.

De nombreux libraires ont demandé à Casterman de pouvoir commander en ligne les bandes dessinées à tout moment afin de satisfaire leur clientèle.

Aussi, Casterman a demandé à ESNEdiPro de mettre en place un site web permettant aux libraires de passer commande en ligne, après s'être authentifiés.

Le catalogue des bandes dessinées doit pouvoir être récupéré en ligne, en accès libre, au format *JSON*, afin que les commerçants puissent l'intégrer automatiquement dans leur site web.

Les données nécessaires de la base de données de *BDPro* étant maintenant en ligne, toutes ces nouvelles applications pourront y accéder à travers une interface de programmation *API REST*.

Récupération du catalogue des bandes dessinées

Le progiciel *BDPro* utilise maintenant une interface de programmation *API REST* pour accéder à la base de données et réaliser toutes les requêtes nécessaires. Pour rappel, l'accès à cette interface de programmation *API REST* est sécurisée et nécessite une authentification.

En ce qui concerne la récupération du catalogue de bandes dessinées au format *JSON*, deux solutions sont possibles :

Solution 1 : modifier l'interface de programmation *API REST* existante pour qu'elle permette cette récupération.

Solution 2 : créer une nouvelle interface de programmation *API REST* d'accès tout public ne permettant que cette récupération.

Après analyse, la seconde solution a été retenue.

Question D2.1

Justifier le choix de la seconde solution du point de vue de la sécurité.

Mission D3 – Application *web* pour les commandes

Une application web vient d'être développée avec l'infrastructure logicielle qui permet la création de programmes grâce à l'environnement de développement (*framework)* nommé *Symfony*. Elle doit permettre aux commerçants de passer commande. Les stocks sont automatiquement gérés et les commandes envoyées à l'entrepôt. L'application *web* exploite aussi une autre interface de programmation *API REST* pour les accès à la base de données en ligne. Cette interface de programmation *API REST* est protégée par une authentification.

Le site permet à tout le monde de consulter le catalogue des bandes dessinées.

Pour avoir le droit de passer commande, un commerçant doit d'abord s'inscrire sur le site (nom, adresse, numéro de téléphone, adresse de messagerie, numéro de Siret, identifiant, mot de passe). Son identité est contrôlée par un commercial de Casterman. Il reçoit alors un courriel lui précisant que son compte est validé.

À partir de là, il peut s'authentifier sur le site à l'aide d'un formulaire lui demandant son identifiant et son mot de passe. Une fois authentifié, il peut gérer son compte (ses informations personnelles), passer commande et suivre l'évolution de ses commandes.

L'authentification des commerçants va être gérée en mémorisant l'identifiant et le mot de passe dans la base de données.

Prise en charge d'un mot de passe oublié

En cas de perte du mot de passe, le commerçant doit pouvoir demander de le réinitialiser. Dans le formulaire d'authentification, un lien est prévu, de type « mot de passe oublié ? ».

Actuellement, il est prévu de demander la saisie de l'adresse de messagerie du commerçant pour la demande de réinitialisation du mot de passe. Un courriel lui est alors envoyé contenant le rappel de son identifiant et un mot de passe qu'il pourra, s'il le souhaite, changer directement sur le site. Le courriel contient aussi le lien pour se rediriger directement vers la page d'authentification.

Ce mode opératoire pose différents problèmes de sécurité.

Question D3.1

a) Identifier les deux problèmes de sécurité existant dans ce mode opératoire.

b) Proposer un autre mode opératoire plus sécurisé.

Documents communs à tous les dossiers

Document commun 1 : Le processus d'édition d'une bande dessinée

Les employés de Casterman ayant la fonction d'éditeur de bandes dessinées, comme Vincent Petit, sont en contact avec le(s) auteur(s) de bandes dessinées. Un auteur est une personne physique qui peut être :

- "auteur complet" signifiant qu'il réalise complètement les bandes dessinées (scénario et dessins) ;
- "scénariste" signifiant qu'il réalise uniquement les scénarii des bandes dessinées ;
- "dessinateur" signifiant qu'il réalise uniquement les dessins des bandes dessinées.

Au cours de l'élaboration d'une bande dessinée, son éditeur échange sur les aspects artistiques avec l'auteur ou les auteurs intervenant lors de la réalisation de la bande dessinée, négocie un contrat avec chaque auteur pour la réalisation de la bande dessinée. Un contrat est établi entre Casterman et un auteur, et concerne une bande dessinée.

Quand un contrat est signé et qu'une bande dessinée est en préparation, son éditeur crée les caractéristiques de la nouvelle bande dessinée dans le progiciel *BDPro* : il lui attribue un numéro ISBN (*International Standard Book Number*) – qui est un identifiant – et renseigne les informations concernant la bande dessinée (titre, format, nombre de pages, etc.) et le ou les auteurs (prénom, nom, adresse, etc.) intervenant dans l'élaboration de la bande dessinée.

Le contrat est signé au format papier entre Casterman et l'auteur concerné par le contrat. Il est ensuite numérisé et stocké sur un serveur de fichiers.

Tous les salariés de Casterman utilisent le progiciel *BDPro* ; les données gérées par ce progiciel sont stockées dans une base de données.

Lorsque la bande dessinée est terminée, elle est numérisée et mise en page par un assistant d'édition au moyen d'un logiciel de publication assistée par ordinateur (PAO). Elle est ensuite éditée dans un document au format *PDF* et transmise à l'imprimeur qui est un sous-traitant situé en Vendée.

Les bandes dessinées imprimées sont ensuite expédiées puis stockées dans un entrepôt situé dans le Loiret (45).

Document commun 2 : Les fonctions des salariés de Casterman

Tous les salariés de Casterman ont accès au progiciel *BDPro*. Les menus et fonctionnalités proposés par le progiciel *BDPro* dépendent de la fonction de l'utilisateur.

En plus de la fonction d'éditeur de bandes dessinées, notamment occupée par M. Petit, d'autres acteurs interviennent dans la réalisation :

- l'assistant d'édition assiste un éditeur de bandes dessinées. Il n'est pas en contact avec les auteurs pour la négociation mais il suit le processus d'édition. Il a moins de droits dans le progiciel *BDPro* qu'un éditeur de bandes dessinées. Il peut apporter des corrections orthographiques aux œuvres, travailler sur la traduction, etc.
- le gestionnaire éditorial : il s'occupe de la gestion, des aspects financiers des bandes dessinées une fois qu'elles sont éditées. Il peut consulter par exemple les chiffres des ventes et la marge commerciale d'une bande dessinée.

Document commun 3 : Extrait de la représentation de la base de données de BDPro

Diagramme de classes :

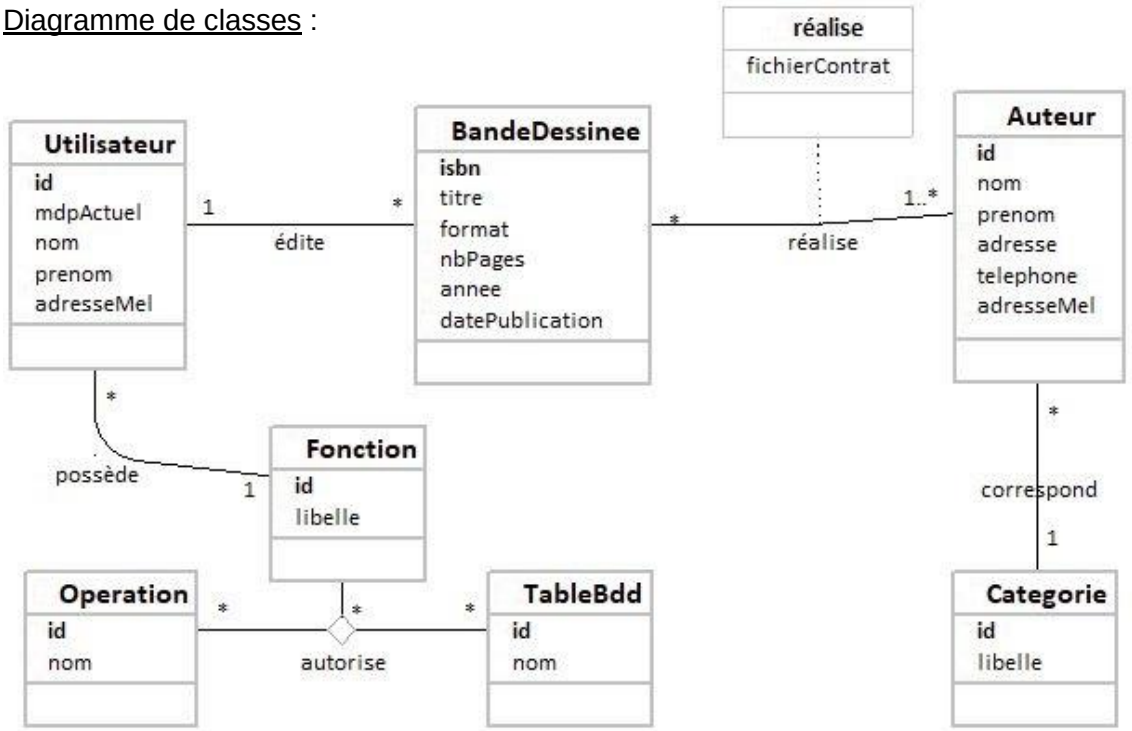


Schéma entité-association :

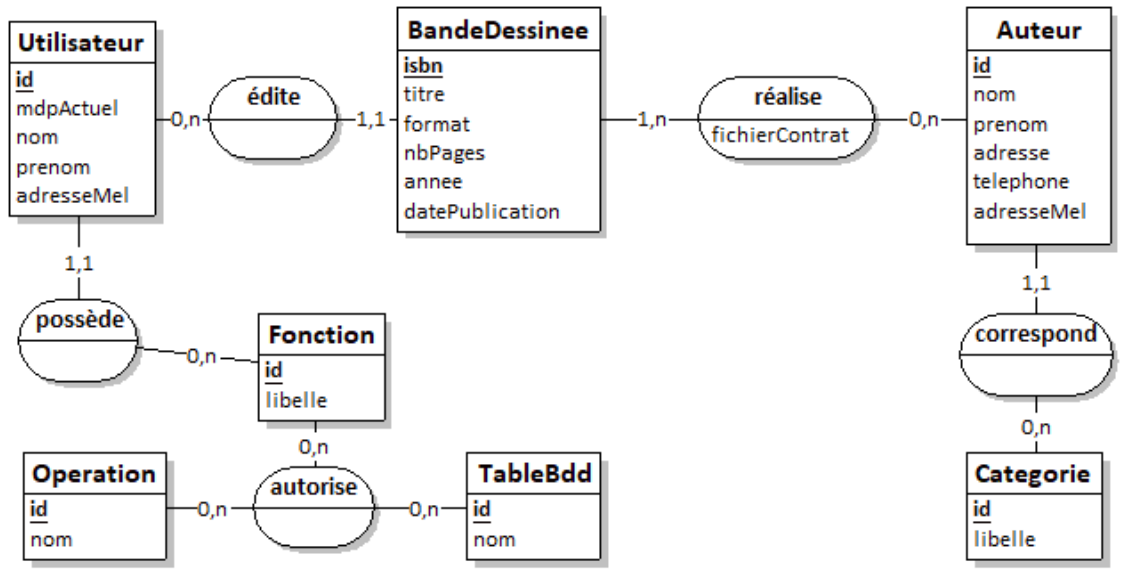


Schéma relationnel :

Utilisateur (id, mdpActuel, nom, prenom, adresseMel, idFonction)

id : clé primaire

idFonction : clé étrangère en référence à id de Fonction

Auteur (id, nom, prenom, adresse, telephone, adresseMel, idCategorie)

id : clé primaire

idCategorie : clé étrangère en référence à id de Categorie

Categorie (id, libelle)

id : clé primaire

BandeDessinee (isbn, titre, format, nbPages, annee, datePublication, idEditeur)

isbn : clé primaire

idEditeur : clé étrangère en référence à id de Utilisateur

Realise (idAuteur, isbn, fichierContrat)

idAuteur, isbn : clé primaire

idAuteur : clé étrangère en référence à id de Auteur

isbn : clé étrangère en référence à isbn de BandeDessinee

Operation (id, nom) // permet de stocker les opérations possibles sur une table de la base de données

id : clé primaire

TableBdd (id, nom) // permet de stocker le nom des tables de la base de données

id : clé primaire

Fonction (id, libelle)

Id : clé primaire

Autorise (idFonction, idOperation, idTableBdd)

idFonction, idOperation, idTableBdd : clé primaire

idFonction : clé étrangère en référence à id de Fonction

idOperation : clé étrangère en référence à id de Operation

idTableBdd : clé étrangère en référence à id de TableBdd

Commentaires :

- isbn : désigne le numéro ISBN d'une bande dessinée,
- fichierContrat : chemin d'accès au fichier du contrat numérisé,
- Categorie contient les catégories d'auteur qui sont "auteur complet", "scénariste" ou "dessinateur".

Document associé au dossier A

Document A1 : Besoins de sécurité pour les récits utilisateurs (user stories) (extraits)

	Intitulé du récit utilisateur (*user story)*	Disponibilité	Intégrité	Confidentialité	Preuve
1	En tant qu'éditeur de bande dessinée, j'enregistre les informations d'une nouvelle bande dessinée et je numérise un contrat passé avec un auteur.	*	**	**	**
2	En tant qu'assistant d'édition, je consulte les informations d'une bande dessinée.	*	**	**	-
3	En tant qu'éditeur de bande dessinée, je transmets une bande dessinée finale au format *PDF* à l'imprimeur.	*	**	**	**

- : sans objet	** : modéré*	*** : important*

Documents associés au dossier B

Document B1 : Modifications et ajouts apportés à la structure de la base de données actuelle

Modification de la table Utilisateur :

Utilisateur (id, mdpActuel**,** dateHeureMdpActuel, nom, prenom, adresseMel, idFonction)
clé primaire : id
dateHeureMdpActuel contient la date et l'heure à laquelle le mot de passe actuel, contenu dans le champ mdpActuel, a été pris en compte pour l'utilisateur
clé étrangère : idFonction en référence à id de Fonction

Ajout d'une nouvelle table qui contiendra l'historique des anciens mots de passe :

HistoMotDePasse (idUtilisateur, dateHeureChangeMdp, motDePasse)
clé primaire : idUtilisateur, dateHeureChangeMdp
clé étrangère : idUtilisateur en référence à id de Utilisateur

Commentaires **:**

- À chaque changement de mot de passe, il faudra ajouter un nouvel enregistrement dans la table HistoMotDePasse.
- Pour chaque utilisateur on ne conservera, dans la table HistoMotDePasse, que les 5 derniers changements de mots de passe.

Document B2 : Déclencheur (trigger) majUtilisateur

Une fonction stockée a été écrite et est utilisée par le déclencheur (*trigger*) :

- *mdpExisteHisto(mdp varchar(60), idUtilisateur int)*

Cette fonction permet de vérifier si un mot de passe existe déjà dans l'historique pour l'utilisateur. La fonction retourne *true* si le nouveau mot de passe a déjà été utilisé, *false* sinon.

```
CREATE TRIGGER majUtilisateur
1. BEFORE UPDATE ON Utilisateur FOR EACH ROW
2.   BEGIN
3.      DECLARE correct BOOLEAN;
4.      SET correct = true;
5.      # la mise à jour concerne-t-elle le changement du mot de passe ?
6.      IF (OLD.mdpActuel != NEW.mdpActuel) THEN
7.         IF (mdpExisteHisto(NEW.mdpActuel,OLD.id) = false) THEN
8.              # le nouveau mot de passe n'existe pas dans l'historique
9.              INSERT INTO HistoMotDePasse
10.                   VALUES (OLD.id, OLD.dateHeureMdpActuel, OLD.mdpActuel);

11.             # INTO affecte à v_nb la valeur retournée par la fonction d'agrégat
12.             SELECT COUNT(*) INTO v_nb
13.                    FROM HistoMotDePasse WHERE idUtilisateur = OLD.id;

14.             DELETE FROM HistoMotDePasse WHERE idUtilisateur = OLD.id
15.                    AND dateHeureChangeMdp = (SELECT MIN(dateHeureChangeMdp)
16.                    FROM HistoMotDePasse WHERE idUtilisateur = OLD.id);
17.        ELSE
18.             SET correct = false; # le mot de passe existe dans l'historique
19.        END IF;
20.     END IF;
21.     IF (correct = false) THEN # message d'erreur
22.         SIGNAL SQLSTATE "45000" SET MESSAGE_TEXT = "mot de passe incorrect";
23.     END IF;
24.   END ;
```

Document B3 : Note fournie par le chef de projet concernant la mémorisation des fichiers au format PDF

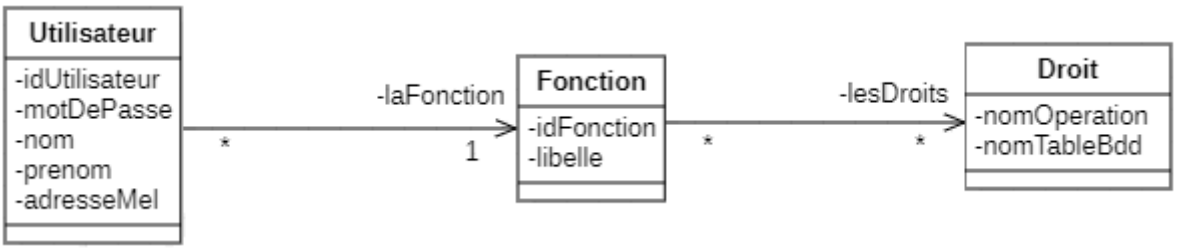
Pour chaque document au format *PDF*, la base de données doit contenir :

- le nom complet de celui-ci, sa date de création ainsi que son chemin d'accès sur le serveur de fichier ;
- les informations permettant de connaître l'utilisateur qui l'a généré et la bande dessinée concernée ;
- la signature électronique de la bande dessinée correspondant au document au format *PDF* retenu comme étant la version finale de la bande dessinée. Cette signature GPG (*GNU Privacy Guard*) est générée par l'éditeur de la bande dessinée.

Pour une question de traçabilité, tous les documents au format *PDF* d'une bande dessinée doivent être mémorisés.

Documents associés au dossier C

Document C1 : Diagramme des classes métier implémentées dans l'application Java (extrait)



Remarque : aucune méthode n'est fournie sur le diagramme de classes. Seules les méthodes utiles sont fournies dans le code des classes métier (document C4).

Document C2 : Classe technique ArrayList (extrait)

```
public class ArrayList<Type>
// Liste de dimension variable contenant des objets de la classe Type.
// Parcours d'une liste de type ArrayList :
for (Type unObjet : uneListe){
   // unObjet est un des objets de la liste uneListe
}
```

Document C3 : Memento SQL

SELECT [**DISTINCT**] { noms-champs | * }
FROM nom-table
[**JOIN** nom-table **ON** critères-de-jointure]
[**WHERE** conditions-de-sélection]
[**GROUP BY** noms-champs-de-groupement]
[**HAVING** conditions-de-sélection-groupe]
[**ORDER BY** noms-de-champs [**ASC** | **DESC**]] ;

Document C4 : Extrait du code Java des classes métier

```
/**
 * Classe métier Utilisateur décrivant un utilisateur de l'application BDPro.
 */
public class Utilisateur {
   private int idUtilisateur;
   private String motDePasse;
   private String nom;
   private String prenom;
   private String adresseMel;
   private Fonction laFonction; // Fonction de l'utilisateur

   // accesseurs et constructeurs codés mais non fournis

   /**
    * Vérifie si l'utilisateur a le droit de réaliser l'opération sur la table.
    * @param uneTable nom de la table
      @param uneOperation nom de l'opération (select, insert, update ou delete)
    * @return un booléen qui indique si l'utilisateur possède le droit demandé
    */
   public boolean possedeDroit(String uneTable, String uneOperation)
   {
      *** PARTIE A COMPLETER ***
   }
}

/**
 * Classe métier Fonction correspondant à la fonction d'un utilisateur.
 */
public class Fonction {
   private int idFonction;
   private String libelle;
   // liste des droits accessibles
   private ArrayList<Droit> lesDroits;
   public ArrayList<Droit> getLesDroits(){ return lesDroits; }
   //autres accesseurs et constructeurs
}

/**
 * Classe métier Droit correspondant à un droit.
 */
public class Droit {
   private String nomOperation; // select, insert, update, delete
   private String nomTableBdd;
   public String getNomTableBdd(){ return nomTableBdd; }
   public String getNomOperation(){ return nomOperation; }
   //autres accesseurs et constructeurs
}
```

Document C5 : Tests unitaires de la méthode possedeDroit

```
class UtilisateurTest {
      Utilisateur unUtilisateur;
      Fonction laFonctionAssistant;
      ArrayList <Droit> lesDroits;
   @Before
   void setUp() throws Exception {
      lesDroits = new ArrayList<Droit>();
      lesDroits.add(new Droit("select","BandeDessinee"));
      lesDroits.add(new Droit("select","Auteur"));
      lesDroits.add(new Droit("select","Realise"));
      lesDroits.add(new Droit("select","Categorie"));
      unUtilisateur = new Utilisateur(1, "ea474f7dcafda10146f1b82b1900cd4c544d3
      fb97a8c55e129a27faa1f2889f9", "Dupont", "Alain", "alain.dupont@gmail.com",
      new Fonction(1,"Assistant d'édition", lesDroits));
   }

   @Test
   void test1() {
      boolean ret = unUtilisateur.possedeDroit("Utilisateur", "select");
      assertFalse("Test sur table incorrecte a échoué", ret);
   }
   @Test
   void test2() {
      boolean ret = unUtilisateur.possedeDroit("Categorie","select");
      assertTrue("Test sur table et opération correctes a échoué", ret);
   }
   @Test
   void test3() {
      boolean ret = unUtilisateur.possedeDroit("Categorie","update");
      assertFalse("Test sur opération incorrecte a échoué", ret);
   }
```

La signature de la méthode *assertFalse* est la suivante :

public static void **assertFalse**(String message, boolean condition)

Si la condition passée en paramètre n'est pas égale à *false* alors assertFalse lève une exception et provoquera le message passé en paramètre.

La signature de la méthode *assertTrue* est la suivante :

public static void **assertTrue**(String message, boolean condition)

Si la condition passée en paramètre n'est pas égale à *true* alors assertTrue lève une exception et provoquera le message passé en paramètre.

Document C6 : Méthode creerEnregLog qui écrit dans le fichier des traces (log)

```
/** Documentation JavaDoc
 * crée un enregistrement dans le fichier des traces (log)
 * @param util l'utilisateur qui effectue l'opération
 * @param droit le droit concerné
 * @param date la date et heure de l'opération
 */
public void creerEnregLog(Utilisateur util, Droit droit, DateTime date){
  SimpleDateFormat forme = new SimpleDateFormat("dd/MM/yyyy-HH:mm:ss");
  String strDate = forme.format(date);
  // mise en forme de l'enregistrement à ajouter dans le fichier log
  String messageLog = "util:" + util.getIdUtilisateur();
  messageLog += ";operation:" + droit.getNomOperation();
  messageLog += ";table:" + droit.getNomTableBdd() +";date:" + strDate;
  // écriture de l'enregistrement dans le fichier log
  logger.info(messageLog);
}
```

Voici un exemple du contenu d'un d'enregistrement du fichier de traces (l*og*) :

`util:5;operation:insert;table:BandeDessinee;date:05/04/2024-08:15:00`

util : est suivi de l'identifiant de l'utilisateur à l'origine de la demande

operation : est suivi du type d'opération réalisée par l'utilisateur (select, insert, update ou delete)

table : est suivi du nom de la table concernée par l'opération

date : est suivi de la date et de l'heure à laquelle l'opération a été réalisée

Document C7 : Fonctions de MySQL de manipulation de dates

- La fonction *current_date()* fournit la date du jour au format YYYY-MM-DD.
  Pour obtenir la date du jour suivie d'une heure précise (8 heures dans l'exemple ci-après), on peut utiliser la fonction *concat* de cette manière : *concat(current_date()," 08:00:00")*.
- La fonction *sysdate()* fournit la date et l'heure courante au format YYYY-MM-DD HH:MM:SS.
- La fonction *datestampdiff(unite, date1, date2)* donne l'écart entre les deux dates passées en paramètre, dans l'unité contenue dans le paramètre *unite*.
  Exemples de valeurs possibles pour le paramètre *unite* : YEAR, MONTH, HOUR, MINUTE

Si date1 est antérieure à date2 alors la fonction retourne un nombre positif, sinon la valeur retournée est un nombre négatif.

Exemple : *datestampdiff*(HOUR,'2024-05-15 8:00:00', '2024-05-15 11:00:00') retourne 3.

Document C8 : Structure et extrait du contenu de la table Log

Nom	Type
id	int(11)
idUtilisateur	int(11)
operation	varchar(6)
nomTable	varchar(30)
dateHeure	datetime

id	idUtilisateur	operation	nomTable	dateHeure
65	5	insert	BandeDessinee	2024-04-05 08:15:00
66	5	select	Auteur	2024-04-05 08:20:30
67	6	select	Categorie	2024-04-05 08:21:00
68	5	delete	Auteur	2024-04-05 08:30:00

Documents associés au dossier D

Document D1 : Description de la structure de l'interface de programmation API

L'interface de programmation API contient les fichiers suivants :

- *.htaccess* : définit les *routes* acceptées et redirige vers *ApiCasterman.php* ;
- *ApiCasterman.php* : récupère les variables passées dans l'adresse (URL) ou dans le corps de la requête et appelle la méthode de la classe Controle correspondant au verbe HTTP reçu (GET, POST, PUT, DELETE) ;
- *Controle.php* : contient la classe Controle qui demande à la classe AccessBdd d'exécuter des requêtes et retourne le résultat ;
- *AccessBdd.php* : contient la classe AccessBdd qui dispose de 5 méthodes :
  - La méthode *select* permet de réaliser des requêtes de sélection de données.
  - La méthode *delete* permet de réaliser des requêtes de suppression.
  - La méthode *update* permet de réaliser des requêtes de mise à jour de données.
  - La méthode *insert* permet de réaliser des requêtes d'ajout de données.

  Les quatre méthodes ci-dessus renvoient un tableau lorsque la requête a réussi et « null » dans le cas contraire.
  - La méthode *autoriseAction* qui permet d'indiquer si un utilisateur peut réaliser une opération sur une table.

Document D2 : Méthode autoriseAction de la classe AccessBdd

La classe AccessBdd contient la méthode suivante :

```
public function autoriseAction(string $idUtil, string $operation,
                                string $tableBdd) : boolean
```

qui retourne vrai si l'utilisateur $idUtil possède le droit de réaliser l'opération $operation sur la table $tableBdd, faux sinon.

Document D3 : Classe Controle de l'interface de programmation API

```
<?php
include_once('AccessBdd.php');
/**
 * reçoit et traite les demandes
 */
class Controle{
    private AccessBdd $accessBdd;
    // Tableau associatif (clé-valeur) contenant le message associé
    // à chaque code de statut HTTP
    private array $tabStatut = array(
                    200 => 'OK',
                    201 => 'Ressource créée',
                    204 => 'Pas de contenu',
                    400 => 'Requête invalide',
                    403 => 'Opération non permise',
                    404 => 'Page non trouvée',
                    500 => 'Erreur serveur');

    /**
```

```
* Constructeur : récupération de l'instance d'accès à la BDD
*/
public function __construct(){
    try{
        $this->accessBdd = new AccessBdd();
    } catch(Exception $e){
        $this->reponse(500, []);
        die();
    }
}

 /**
 * réponse renvoyée (affichée) au client au format JSON
 * @param int $code statut de la requête
 * @param array|null $result résultat de la requête
 */
private function reponse(int $code, array|null $result): void{
    if ($result == null){
        $code = 400;
    }
    $retour = array( 'code' => $code,
                     'message' => $this->$tabStatut[$code],
                     'result' => $result
            );
    echo json_encode($retour, JSON_UNESCAPED_UNICODE);
}

/**
 * requête HTTP de type GET (select)
 * @param string $table nom de la table
 * @param array $champs nom et valeur des champs critères de recherche
 */
public function get(string $table, array $champs): void {
    $result = $this->accessBdd->select($table, $champs);
    $this->reponse(200, $result);
}

/**
 * requête HTTP de type DELETE
 * @param string $table nom de la table
 * @param array $champs nom et valeur des champs critères du delete
 */
public function delete(string $table, array $champs): void {
    $result = $this->accessBdd->delete($table, $champs);
    $this->reponse(204, $result);
}



/**
```

```
     * requête HTTP de type POST (insert)
     * @param string $table nom de la table
     * @param array $champs nom et valeur des champs à insérer
     */
    public function post(string $table, array $champs): void {
        $result = $this->accessBdd->insert($table, $champs);
        $this->reponse(201, $result) ;
    }

     /**
     * requête HTTP de type PUT (update)
     * @param string $table nom de la table
     * @param string $id valeur de l'id de la ligne à modifier dans la table
     * @param array $champs nom et valeur des champs à modifier
     */
    public function put(string $table, string $id, array $champs): void {
        $result = $this->accessBdd->update($table, $id, $champs);
        $this->reponse(200, $result);
    }
}
```

Remarques :

- Le paramètre $champs est un tableau associatif de type clé-valeur. La **clé** de ce tableau est le nom de la colonne de la table concernée par la requête, la **valeur** est la valeur de cette colonne.

  Deux exemples d'utilisation de ce tableau associatif :

```
$champs['nom'] = "Dupont";  // assigne la valeur Dupont à la clé nom
$val = $champs['nom']       // val reçoit la valeur de l'élément
                            // dont la clé est nom
```

- En plus des champs nécessaires à l'opération, le tableau $champs contient toujours un élément dont la clé est idUtilisateur et sa valeur l'identifiant de l'utilisateur qui demande la requête HTTP de l'interface de programmation d'application (API).