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

U6 – CYBERSÉCURITÉ DES SERVICES INFORMATIQUES

SESSION 2024

_____

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.

GSIC

Barème

DOSSIER A	**Préparation de la mise en place d'une authentification unique (SSO)**	20 points
DOSSIER B	**Mise en place du service SSO pour le système de gestion administratif**	35 points
DOSSIER C	**Adaptation de la politique de sécurité de l'application mobile *Rescousse***	25 points
	TOTAL	80 points

Dossier documentaire

Présentation du contexte

Un Service départemental d'incendie et de secours (Sdis) est un établissement public administratif. Les Sdis ont différentes missions de prévention, de protection et de lutte contre les incendies, les risques de sécurité civile, etc.

Le Sdis situé dans la ville de M. est équipé de 504 engins et a réalisé 43 632 interventions en 2023.

Les 1 723 sapeurs-pompiers volontaires et 484 sapeurs-pompiers professionnels sont répartis dans les 71 centres d'incendie et de secours du département. Le personnel administratif et technique spécialisé comprend 112 agents.

Le **groupement des systèmes d'information et communication** (**Gsic**) du Sdis de la ville de M. compte un effectif de 26 agents et est composé de plusieurs services dont le service informatique, le service de cartographie et le service d'analyse décisionnelle.

Le **service informatique** a un effectif de 16 agents et assure l'ingénierie, la maintenance et la sécurité des systèmes de gestion administratif et opérationnel.

Le **système de gestion opérationnel** assure la gestion des interventions, de la prise des appels d'urgence jusqu'au retour à la caserne des équipes d'intervention. Il comporte 180 postes informatiques et 55 serveurs. Il s'agit d'un système à très haute disponibilité, intégré à l'infrastructure téléphonique et radio. Ce système comprend notamment une application mobile *Rescousse* fonctionnant sur tablette numérique de type *Android*. Elle permet aux sapeur-pompiers en intervention de transmettre en temps réel les premières informations concernant les victimes aux services hospitaliers par une interface de programmation d'application (API).

Le **système de gestion administratif** comprend différents logiciels spécialisés pour gérer les ressources humaines, la formation, le suivi médical, la logistique, la prévention et la prévision ainsi que la section de jeunes sapeurs-pompiers et les engagés civiques.

Ces logiciels sont utilisés par les agents du Sdis à partir de 560 postes de travail. Le cœur du système de gestion administratif se compose de 25 serveurs répartis dans deux locaux techniques. Par ailleurs, 16 serveurs secondaires sont affectés aux plus importants centres d'incendie et de secours du département.

Politique de sécurité des systèmes d'information

Avec ses nombreux utilisateurs et terminaux mobiles, le Sdis n'échappe pas à l'ingénierie sociale. Conscient que le facteur humain est une clé de la cybersécurité et soucieux d'alléger le poids administratif des sapeurs-pompiers, le responsable de la sécurité des systèmes d'information (RSSI) a adopté une démarche de sécurité positive. Il ne s'agit pas de faire peur aux agents avec des discours anxiogènes, mais de les accompagner tout en améliorant le niveau de sécurité.

Un budget a été dégagé afin de mettre en œuvre une politique de sécurité des systèmes d'information (PSSI). Le plan associe :

- des actions de sensibilisation et de formation à la cybersécurité inscrites dans la durée ;
- des modifications des logiciels destinées à faciliter leur utilisation, à simplifier le travail administratif et à améliorer la sécurité.

M. Dinant, maitre d'œuvre (MOE) au service informatique participe à la mise en œuvre de ce plan et est chargé :

- de la mise en place d'un service d'authentification unique (*Single Sign On* – SSO) sur le parc applicatif existant du système de gestion administratif ;
- de la mise en place de la politique de sécurité dans l'application mobile *Rescousse*.

Élodie, votre collègue, et vous-même faites partie de l'équipe de M. Dinant et participez à ces différentes missions. Vous vous appuierez sur le dossier documentaire mis à votre disposition.

Dossier A – Préparation de la mise en place d'une authentification unique

Le système d'information (SI) comprend différentes applications, chacune ayant sa propre manière de gérer les utilisateurs, les droits et les habilitations. Afin de mettre en place l'authentification unique (*Single Sign-On* - SSO), il est nécessaire de centraliser la gestion des utilisateurs et des rôles pour les rendre communs aux différentes applications.

Dans un premier temps, il convient de faire l'inventaire de l'existant et vous participez à cette étape.

Mission A1 – Fichiers actuels de journalisation des authentifications

Chaque application du système de gestion administratif dispose d'une gestion des utilisateurs et d'un système de traçabilité des authentifications qui lui est propre ; pour cela, elle utilise un fichier de journalisation.

La procédure d'enregistrement, de sauvegarde et d'archivage, mise en place pour les fichiers de journalisation des authentifications des différentes applications, est la suivante :

- Toute authentification, ou tentative d'authentification, est enregistrée dans le fichier de journalisation de l'application concernée, sur l'un des serveurs du Sdis.
- Si nécessaire, les données peuvent être extraites des fichiers de journalisation à l'aide de filtres.
- Les fichiers de journalisation, faisant partie des données système, sont sauvegardés tous les jours sur des disques externes et les serveurs disposent d'un système de réplication en miroir (RAID 1 - *mirroring*).
- Une fois par mois, ces fichiers de journalisation sont transférés sur bandes magnétiques, via le réseau interne du Gsic, afin d'être archivés. Avant leur transfert, il y a compression avec chiffrement, mais sans vérification des sommes de contrôles (empreinte numérique). Les journaux archivés sont conservés pendant 6 mois.

Question A1.1
Rappeler l'objectif réglementaire de la traçabilité des accès.

Question A1.2
a) Indiquer la différence d'objectif entre l'archivage et la sauvegarde.
b) Expliquer pour quelle raison les fichiers de journalisation des authentifications sont archivés.

M. Dinant s'interroge sur la procédure mise en place en matière de confidentialité et d'intégrité de l'archivage des journaux exigées par la loi.

Question A1.3
Préciser, en le justifiant, si la confidentialité et l'intégrité des fichiers de journalisation sont garanties lors du processus d'archivage.

Mission A2 – Inventaire des comptes d'authentification existants

Étant donné que chaque application dispose de son propre système de gestion de l'authentification, les caractéristiques des utilisateurs et de leurs rôles sont enregistrées dans plusieurs bases de données.

Avant la mise en place de l'authentification unique, il convient de faire un inventaire des comptes actuels. Un programme *InventaireHabil* sera développé à cet effet. Il exploitera une vue en entrée et alimentera la nouvelle base de données *MySql* nommée BdInventaire.

Vous devez écrire la vue *v_liste_comptes* de la base BdInventaire, qui permettra d'obtenir les caractéristiques des utilisateurs enregistrés dans les différentes bases de données *MySql* utilisées par les applications.

La vue aura la structure suivante : **v_liste_comptes** (origine, nom, prenom, compte_login, roles)

La colonne origine contient le nom de la base de données d'où proviennent les caractéristiques de l'utilisateur. Une donnée inexistante dans une base sera remplacée par le mot clé NULL.

Question A2.1
Écrire la vue *v_liste_comptes*.

Dans les bases de données Personnel et Logistique, un même compte utilisateur peut posséder plusieurs rôles applicatifs. La liste des rôles est enregistrée dans un seul champ où ils sont séparés par des virgules.

Le programme *InventaireHabil* insèrera les données retournées par la vue *v_liste_comptes* dans les tables de la base BdInventaire. Il déterminera l'application du compte utilisateur à ajouter à partir de l'origine des données. Un identifiant sera attribué à chaque nom de rôle qui n'est rattaché qu'à une seule application.

Le programme *InventaireHabil* utilisera le compte *MySql prep_sso* du poste local.

Question A2.2
Compléter la modélisation de la base BdInventaire afin que le programme *InventaireHabil* puisse enregistrer cet inventaire.

Question A2.3
a) Créer le compte *prep_sso*.
b) Attribuer les droits strictement nécessaires au compte pour le programme *InventaireHabil*.

Dossier B – Mise en place du service SSO pour le système de gestion administratif

Mission B1 – Renforcer la sécurité des données personnelles et sensibles

Après la mise en place du service SSO, un utilisateur n'aura qu'un seul compte qui pourra donner accès à de très nombreuses données, dont des données personnelles et sensibles. La PSSI prévoit plusieurs mesures parmi lesquelles des actions de sensibilisation, l'application stricte de la charte informatique et le renforcement de la sécurité en chiffrant les données personnelles et sensibles dans les bases de données.

Lors d'un atelier de sensibilisation à la sécurité du système d'information que vous animez, un membre du service de gestion administratif vous avoue avoir divulgué ses identifiants à un de ses stagiaires, ce qui a permis au stagiaire de se connecter à l'intranet.

Question B1.1
a) Expliquer à votre interlocuteur la ou les infractions qu'il a commises.
b) Préciser si le stagiaire est en faute.
c) Décrire les risques éventuellement encourus par les deux personnes.

Question B1.2
Préciser à quelle condition règlementaire la charte informatique peut s'imposer aux employés.

Les différentes contraintes opérationnelles de l'activité des sapeurs-pompiers les exposent à de hauts risques sanitaires. Ils sont donc scrupuleusement suivis sur le plan médical. Leur aptitude médicale est évaluée lors d'une visite médicale d'admission au moment du recrutement, annuellement lors d'une visite médicale de maintien en activité et également lors des visites de reprise ou des visites occasionnelles.

L'application de gestion des visites médicales utilise deux bases de données *SQL Server* nommée Medical et DossierMedical. Elles ont déjà fait l'objet de mesures de sécurité parmi lesquelles :

- La pseudonymisation des patients pour isoler dans des bases séparées les données de santé détaillées et les données d'identification. Une clé anonyme propre à chaque patient permet de calculer l'identifiant du patient codé correspondant. C'est le rapprochement de ces 2 clés qui permet d'inverser la pseudonymisation.
- La protection de la base DossierMedical avec le chiffrement transparent des données (TDE). TDE assure un chiffrement et un déchiffrement des données et des fichiers journaux en entrées/sorties et en temps réel. La grande majorité des données médicales sont stockées dans cette base DossierMedical.

Pour des raisons de performance, la base Medical n'avait pas été chiffrée. Afin de limiter l'impact sur les performances, le MOE M. Dinant, en accord avec le comité technique, a décidé de chiffrer uniquement les colonnes correspondant à des données personnelles et sensibles. Ces colonnes feront ainsi l'objet d'un chiffrement symétrique avec certificat dans la base *SQL Server* à l'aide d'instructions Transact-SQL.

Question B1.3
Dresser la liste des données sensibles de la base de données Medical.

Question B1.4
Écrire un scénario de risque qui rende pertinent le chiffrement des données de la base en considérant que les accès par l'application sont sécurisés.

M. Dinant vous demande de participer à la mise en place du chiffrement des données personnelles et sensibles, au niveau de la base de données Medical sans modifier les applications qui l'utilisent. Un scénario de la mise en œuvre a été décrit ; il utilise des vues en langage SQL qui seront nommées exactement comme les tables actuelles. Les données seront chiffrées dans les tables et seront déchiffrées dans la vue. Seuls les utilisateurs habilités pourront utiliser les vues via une authentification par certificat, les accès seront journalisés. Un utilisateur non habilité qui voudrait lire les données ne verrait que des données chiffrées.

Question B1.5
Écrire les requêtes permettant de tester le point 4 du scénario pour la table **PatientAnonyme** sur le champ *rhesus*.

Mission B2 – Audit de sécurité et intégration

La PSSI a prévu la mise en place d'un serveur de fédération d'identité afin de fournir un service d'authentification unique. Un prototype de serveur SSO a été installé sur une machine à l'aide d'une solution libre qui respecte le protocole d'authentification *OpenId Connect* permettant de travailler à l'aide d'échanges de jetons JSON *web token* (JWT) entre les applications nécessitant une authentification et le serveur SSO. Le prototype a été baptisé *Fede-Gsic*.

M. Dinant demande de réaliser quelques tests pour intégrer le service *Fede-Gsic* dans les applications existantes. La première application concernée est l'application de gestion des formations du personnel. Il s'agit d'un portail internet appelé *Sdisform*.

Pour qu'une application accède à *Sdisform*, il faut utiliser une interface de programmation (Application Program Interface - API) développée en Java.

Élodie a effectué les tests d'analyse de cette interface de programmation. Elle a écrit un courriel en guise de compte-rendu.

Question B2.1
Identifier les conséquences que cela pourrait avoir en matière de sécurité des traitements proposés dans l'interface de programmation.

Question B2.2
Proposer à Élodie un moyen pour résoudre le problème qu'elle rencontre avec la modification du public concerné en lui détaillant ce qui pose précisément problème et pourquoi.

Élodie a débuté son travail d'intégration du service SSO dans l'interface de programmation et le programme qui l'utilise. Elle rencontre un nouveau souci lors de ses tests de l'interface. Elle a pu obtenir un jeton JWT en s'authentifiant auprès du serveur SSO et après avoir fait sa pause-café durant 15 minutes, elle essaie d'interroger l'interface avec le jeton précédemment obtenu et obtient l'erreur suivante :

```
{
    "error": "invalid_token",
    "error_description": "Token verification failed"
}
```

Question B2.3
Émettre l'hypothèse la plus probable concernant la nature du problème amenant à cette situation.

Afin de gérer l'expiration du jeton JWT initial, une seconde méthode *getToken* sans paramètre doit être ajoutée dans la classe *AuthentificationUtils.* Cette nouvelle méthode mettra à jour les jetons à l'aide du jeton de rafraichissement actuel et retournera le jeton rafraichi.

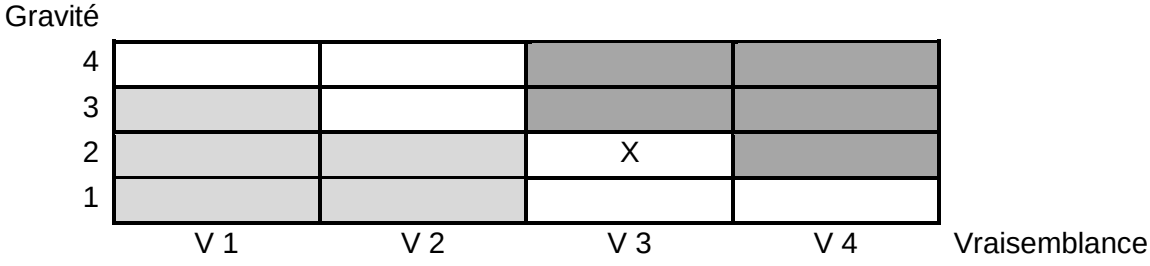
Élodie a écrit les spécifications de cette méthode.

Question B2.4
Compléter cette nouvelle méthode.

Mission B3 – Impact de l'authentification unique sur les risques établis

Une analyse de risques avait été réalisée pour cette application de gestion des formations lors de sa réalisation il y a quelques années.

Scénario de risque : un sapeur-pompier se fait dérober son mot de passe et un pirate l'utilise pour se connecter sur l'application d'inscription aux formations.



Commentaires : ce scénario est assez probable. Il arrive fréquemment que des utilisateurs se fassent usurper leur compte sur une application en ligne. Cependant, les conséquences sont minimes sur le système d'information et les administrateurs pourront rapidement rétablir les fausses informations saisies par le pirate et réinitialiser le mot de passe pour arrêter l'usurpation.

Question B3.1
a) Justifier en quoi la mise en place du service SSO pourrait modifier ce scénario.
b) Proposer une piste d'amélioration pour minimiser le niveau du risque.

Dossier C – Adaptation de la politique de sécurité de l'application mobile *Rescousse*

Mission C1 – Renforcement de la sécurité d'accès

L'application mobile *Rescousse* est installée sur les tablettes *Android* itinérantes et les pompiers l'utilisent lors des interventions pour transmettre des informations aux centres hospitaliers afin d'anticiper la prise en charge des victimes.

L'accès aux tablettes est verrouillé par un code PIN à 4 chiffres. Il est modifié chaque début de mois automatiquement par un logiciel d'administration à distance. Pour des raisons de praticité lors des interventions, les tablettes se verrouillent automatiquement après 15 minutes d'inactivité. L'accès aux paramètres (*Settings*) des tablettes et l'installation d'application sont bloqués pour les utilisateurs. Les données contenues dans la tablette sont chiffrées.

L'application *Rescousse*, traitant des données personnelles et sensibles, requiert bien sûr une authentification auprès d'un serveur d'authentification distant. Cette authentification est systématiquement précédée par la redemande du code PIN à 4 chiffres pour accéder à l'application.

Afin d'améliorer la sécurité, la PSSI a prévu de renforcer le système de déverrouillage. Le temps d'inactivité de la tablette avant verrouillage automatique sera réduit à 10 minutes. À chaque accès à la tablette, ainsi qu'à chaque accès à l'application *Rescousse*, l'utilisateur pourra choisir soit de saisir un mot de passe de 8 caractères, soit d'utiliser la biométrie, plus précisément, la reconnaissance faciale en remplacement du code PIN à 4 chiffres.

Question C1.1
Préciser contre quel type d'attaque un mot de passe à 8 caractères est plus protecteur qu'un code PIN à 4 chiffres.

Question C1.2
Expliquer en quoi la reconnaissance faciale se distingue d'un mot de passe d'un point de vue juridique.

Dans un premier temps, la PSSI prévoit de mener une expérimentation dans une caserne avec un dispositif biométrique dont le gabarit est stocké dans l'appareil. L'administrateur des équipements itinérants se chargera de faire enregistrer les modèles du visage des pompiers volontaires pour cette expérimentation. Un journal des accès à la tablette, mis à jour par l'application *Rescousse* sera exploité afin de faire le point et de passer éventuellement à l'étape suivante de l'expérimentation.

Une version supplémentaire de l'application *Rescousse* va être gérée, à destination de la caserne expérimentale. Les modifications prévues sont :

- accès à l'application au choix par biométrie ou mot de passe avec une limite à trois essais ;
- journalisation des accès à l'application ;
- remontée de la journalisation sur un serveur du Sdis.

Au moyen de la documentation *Android*, votre collègue Elodie a récupéré et rassemblé le code mettant en œuvre l'authentification biométrique. Elle a créé l'activité *LoginBiometrieActivity* et défini les classes EntreeLog et DbLogRescousse à utiliser pour enregistrer le journal des accès dans une base de données SQLite. Vous devez compléter le code de ces classes.

Question C1.3
Écrire le code de la méthode *ajouterEntreeLog* de la classe DbLogRescousse.

Il reste à compléter le code de la classe *LoginBiometrieActivity* pour écrire dans le journal des accès. La valeur du type à renseigner est *LoginBiometrie*.

Les autres données du journal sont à renseigner de la façon suivante :

Authentification	Résultat	Message
En cas de succès	Succes	"OK"
En cas d'erreur	Echec	code erreur – texte erreur

Question C1.4
Compléter le code de la méthode *onCreate* de la classe *LoginBiometrieActivity*.

Il est prévu une remontée de la journalisation des accès sur un serveur du Sdis. Ceux-ci seront enregistrés en base de données dans une table de la structure suivante :

LogRescousse (adMAC, idEntreeLog, date, heure, type, resultat, message)

adMAC, idEntreeLog : clé primaire

adMAC : adresse MAC de la tablette d'où provient l'entrée log

type : type d'authentification('LoginBiometrie' ou 'LoginMdp')

resultat : résultat de la tentative de connexion ('Succes' ou 'Echec')

message : libellé de l'erreur en cas d'échec de connexion

Question C1.5
a) Préciser si cette table contient toutes les données indispensables à une journalisation, en vous appuyant sur les recommandations de la CNIL.
b) Indiquer la durée recommandée par la CNIL pour la conservation de ces données de journalisation.

M. Dinant vous demande d'analyser les données de la table de journalisation. Il aimerait connaitre le nombre de tentatives de connexion en échec pour chaque type de connexion et pour chaque message d'erreur.

Question C1.6
Donner la requête qui permet d'obtenir les informations demandées par M. Dinant.

Documents associés au dossier A

Document A1 : Nom des bases de données MySql et extrait des tables contenant des comptes utilisateurs applicatifs

Pour chaque table, *id* est la clé primaire.

Personnel : Base de données de la gestion des ressources humaines

Compte_Employe(id, login, password, salt, matricule, nomUser, prenomUser, rolesUser, telephonePro, telephonePri, ville, dateCreation, statut, mail, fonction)

Formation : Base de données de la gestion de la formation

Utilisateur(id, compte, mot_passe, sel, nom, prenom)

Logistique : Base de données de la gestion de la logistique

Compte(id, compte, mot_passe, nom_compte, prenom_compte, roles_compte, ville_compte)

Prevention : Base de données de la gestion de la prévention

User(id, login, password, salt, matricule, nom, prenom, role, telephoneProf)

Document A2 : Extraits de la documentation MySQL

CREATE USER : permet la création d'un nouveau compte utilisateur MySQL.
`CREATE USER 'MorineauJ'@'localhost' IDENTIFIED BY 'unMotDePasse';`
`-- création d'un utilisateur 'MorineauJ' qui se connecte depuis le serveur`
`-- local avec le mot de passe 'unMotDePasse'.`

DROP USER : permet la suppression d'un compte utilisateur MySQL. `DROP USER 'MorineauJ'@'localhost'; -- supprime l'utilisateur 'MorineauJ'`

GRANT : permet l'attribution de droit à un compte utilisateur.
`GRANT permission,… ON nom_objet TO utilisateur;`
`-- permission vaut DELETE, INSERT, REFERENCES, SELECT, UPDATE, ALTER, ALL ou`
`-- CREATE {DATABASE|DEFAULT|FUNCTION|PROCEDURE|CERTIFICATE|RULE|VIEW|TABLE}`

CREATE VIEW : permet la création de vue.
`CREATE VIEW view_name [(column_list)] AS select_statement [WITH CHECK OPTION]`
`CREATE VIEW uneVue AS SELECT * FROM test.uneTable; -- crée la vue uneVue`
`-- affichant les informations de la table uneTable de la base de données test`

UNION : opérateur ensembliste permettant de combiner dans un résultat unique des lignes provenant de deux à plusieurs ordres SELECT qui retournent le même schéma
`SELECT colonne8, colonne2 FROM table1`
`UNION`
`SELECT colonne2, colonne5 FROM table2`

Document A3 : Modélisation de la base BdInventaire

Diagramme de classe

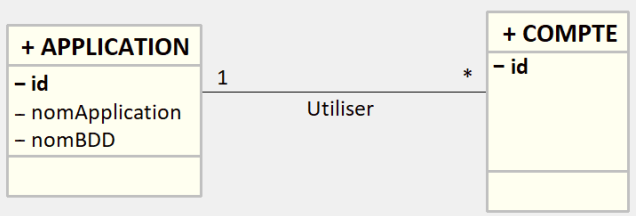
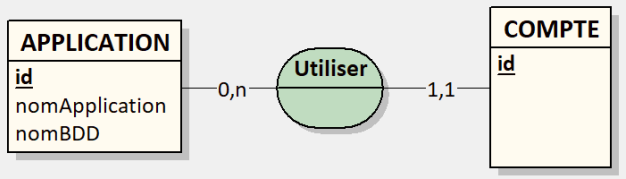


Schéma entité-association



Documents associés au dossier B

Document B1 : Extraits de la charte informatique actuelle (avant mise en place du service SSO)

Section 1 Champ d'application

Section 1.1 Personnes concernées

La présente charte informatique s'applique à l'ensemble des individus qui utilisent les systèmes d'information du Sdis de la ville de M., y compris, mais sans s'y limiter :

- les salariés de l'organisation ;
- les intérimaires engagés pour des missions au sein de l'organisation ;
- les stagiaires exerçant leurs missions au sein de l'organisation ;
- les employés de sociétés prestataires qui accèdent aux systèmes d'information dans le cadre de leurs fonctions ;
- les visiteurs occasionnels ayant obtenu des droits d'accès temporaires ;
- de manière générale, toute personne ayant obtenu des droits personnels d'utilisation des systèmes d'information.

Dans l'ensemble de ce document, ces personnes seront désignées sous le terme "utilisateur".

[…]

Section 2.1 : Accès et utilisation des ressources du système d'information

L'accès et l'utilisation des ressources du système d'information du Sdis de la ville de M. sont strictement réglementés et nécessitent une autorisation préalable.

2.1.1 Codes d'accès

Chaque utilisateur peut se voir attribuer un ou plusieurs codes d'accès aux ressources du système d'information, composés des éléments suivants :

- un identifiant unique attribué par le Sdis de la ville de M. ;
- un mot de passe choisi par l'utilisateur, tout en respectant les règles en vigueur au sein du Sdis de la ville de M.

2.1.2 Responsabilités de l'utilisateur

Afin de garantir la sécurité des accès aux ressources du système d'information, l'utilisateur s'engage à assurer :

1. **Confidentialité des codes d'accès :** Garder strictement confidentiel(s) son (ses) code(s) d'accès et ne jamais les communiquer à un tiers. L'utilisateur est responsable de toute utilisation de ses codes d'accès, et sa responsabilité pénale et civile peut être engagée en cas de divulgation volontaire à un tiers. Le stockage des codes d'accès n'est autorisé qu'avec le coffre-fort de mots de passe mis à disposition par le Sdis de la ville de M.
2. **Non utilisation des codes d'autrui :** Ne pas utiliser les codes d'accès d'un autre utilisateur et ne pas chercher à les découvrir.
3. **Respect des droits d'accès :** S'abstenir d'accéder ou de tenter d'accéder à des ressources du système d'information en contournant les droits attribués.

[…]

Section 5.1 : Sanctions

La présente charte informatique revêt une portée juridique contraignante pour les utilisateurs. Tout manquement aux règles et mesures de sécurité énoncées dans cette charte peut entraîner des sanctions, proportionnées à la gravité des faits, et pouvant inclure :

- Avertissements : L'utilisateur peut recevoir un avertissement formel en cas de violation mineure.
- Limitations ou suspensions : L'accès à tout ou partie du système d'information et de communication peut être limité ou suspendu en cas de violation plus grave.
- Sanctions disciplinaires : Des mesures disciplinaires conformes aux procédures établies dans le règlement intérieur et le Code du travail peuvent être appliquées en cas de violations graves et répétées.

En outre, le Sdis de la ville de M. se réserve le droit d'engager des poursuites pénales et/ou civiles, notamment en cas de fraude informatique, de non-respect des droits d'auteur ou de violation du secret des correspondances.

Le responsable de la sécurité des systèmes d'information peut supprimer ou isoler et conserver toute trace de logiciels, progiciels, programmes ou fichiers créés ou introduits dans le système d'information, en violation des droits des tiers, sans préjudice de l'application de sanctions.

Document B2 : Extraits de la documentation SQL Server

Le langage *Transact-SQL* (*T-SQL*) est une extension propriétaire de *Sybase* et *Microsoft* au langage *SQL*.

Exemples (Transact-SQL)

```
-- Permet d'ajouter un certificat à une base de données dans SQL Server.
CREATE CERTIFICATE Medical18
   WITH SUBJECT = 'Visites medicales';
GO
-- Crée une clé symétrique avec certificat et algorithme AES_256
CREATE SYMMETRIC KEY Medical_Key
    WITH ALGORITHM = AES_256  ENCRYPTION BY CERTIFICATE Medical18;
GO
-- Retourne le numéro de sécurité sociale chiffré, de tous les patients.
SELECT EncryptByKey(key_GUID('Medical_Key'), num_secu) FROM Patient ;
```

Type de données pour les données chiffrées

varbinary [(*n* | max)]
Données binaires de longueur variable. n peut être une valeur comprise entre 1 et 8 000.

Syntaxe

```
-- Permet d'ajouter, de modifier ou de supprimer une colonne dans une table.
ALTER TABLE nom_table {ADD | ALTER COLUMN | DROP} nom_colonne [type];
-- Procédure stockée qui permet de renommer un objet, une table, une colonne…
EXEC sp_rename 'ancien_nom', 'nouveau_nom'[, 'type de l'objet']
-- 'type de l'objet' prend ses valeurs dans {COLUMN, DATABASE, INDEX, … }
--dans le cas d'une table, le 'type de l'objet' n'est pas attendu
--dans le cas d'une colonne, 'ancien_nom' prend la forme nom_table.nom_colonne
```

Document B3 : Extrait de la base de données Medical

Patient (id, num_secu, genre_biologique, nom, prenom, date_naissance, pays_naissance, situation_maritale, tel_priv, tel_prof, mail_prof, rue, complement_rue, code_postal, ville, id_fonction, id_service, cle_anonyme)

id : clé primaire
id_fonction : clé étrangère en référence à id de Fonction
id_service : clé étrangère en référence à id de Service

PatientAnonyme (id_anonyme, rhesus, id_dossier)

id_anonyme : clé primaire
id_dossier : clé étrangère en référence à id de Dossier (table non présentée ici)

Visite (id_ anonyme, date, taille, poids, masse_graisseuse, masse_musculaire, tension, profil, id_dossier_visite)

id_ anonyme, date : clé primaire
id_ anonyme : clé étrangère en référence à id_anonyme de PatientAnonyme
id_dossier_visite: clé étrangère en référence à id de DossierVisite (table non présentée ici)

Vaccination (id_anonyme, date, id_vaccin)

id_ anonyme, date : clé primaire
id_ anonyme : clé étrangère en référence à id_anonyme de PatientAnonyme

Vaccin (id, nom)

id : clé primaire

Fonction (id, libelle)

id : clé primaire

Service (id, nom)

id : clé primaire

Informations sur les données

Abréviations : tel (téléphone), priv (privé), prof (professionnel), num_secu (numéro sécurité sociale)
Rhésus : Le système Rhésus est un système de groupe sanguin porté uniquement par les globules rouges et d'importance majeure. *Remarque : le rhésus sanguin d'un pompier n'est pas considéré comme une donnée sensible, car celui-ci est inscrit en clair sur l'uniforme de certains corps de pompiers.*
Vaccin : Exemples : BCG, hépatite B, hépatite A, leptospirose, …
Profil : L'évaluation médicale, en s'aidant de la cotation des sigles S, I, G, Y, C, O et P, permet la détermination d'un profil médical individuel (S : membres supérieurs, I : membres inférieurs, G : état général, Y : vision, C : sens chromatique, O : audition, P : psychisme). Les missions confiées aux sapeurs-pompiers (volontaire ou professionnel) prennent en compte l'âge et le profil.

Document B4 : Scénario de chiffrement des données personnelles et sensibles de la base de données Medical

1. Faire une sauvegarde de la base de données.
2. Créer un compte *SQL Server* avec les droits pour les opérations ci-dessous.
3. Mettre en place le chiffrement :
   a. créer le certificat,
   b. créer la clé symétrique de chiffrement.
4. Chiffrer les données existantes. Pour chacune des tables Patient, Visite et Vaccination :
   a. ajouter une nouvelle colonne par colonne à chiffrer avec le type *varbinary* de 30 caractères,
   b. renseigner chaque nouvelle colonne avec la donnée chiffrée,
   c. supprimer les colonnes contenant les données en clair,
   d. renommer la nouvelle colonne avec le nom de la colonne supprimée.
5. Rendre le chiffrement transparent pour l'application. Pour chacune des tables Patient et Visite :

a. renommer les tables en T_Patient et T_Visite,

b. créer une vue portant le même nom que la table d'origine (sans chiffrement) qui sera utilisée de manière transparente par l'application à la place de la table,

c. pour l'insertion, la mise à jour et la suppression, créer des déclencheurs (*triggers*) INSTEAD OF posés sur la vue et mettant à jour de façon transparente la table chiffrée.

6. Attribuer aux utilisateurs de l'application, les droits sur la vue et sur la clé de chiffrement.

Document B5 : Extrait des méthodes figurant dans le contrôleur de l'interface de programmation (API)

```
@RestController
public class FormationController {
    // les 2 jetons sont stockés dans l'objet authUser et
    // disponibles depuis toute méthode du controleur
    private AuthentificationUtils authUser; // initialisé dans le constructeur
    private List<Formation> formations = new ArrayList<>();
    private FormationDAO formationsDAO = new FormationDAO();

    @GetMapping("/formations")
    public List<Formation> obtenirFormations() {    }

    @PostMapping("/formations")
    public void creerFormation(@RequestBody Formation formation) {    }

    @DeleteMapping("/formations/{code}")
    public void supprimerFormation(@PathVariable String code) {    }

    @PutMapping("/formations/{code}")
    public void mettreAJourFormation(@PathVariable String code,
            @RequestBody Formation formation) {    }
}
```

Document B6 : Contenu du retour de la requête d'obtention du jeton (token) JWT

Lorsqu'un utilisateur a été correctement identifié sur le serveur *SSO*, un flux au format *JSON* est renvoyé à l'application qui a initié l'authentification.

Dans l'extrait ci-dessous les suites longues de caractères sont remplacées par **…**

```
{
  "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIn0=...",
  "expires_in": 300,
  "refresh_expires_in": 1800,
  "refresh_token": "QnJhdm8sIHZvdXMgYXZleiBkw6ljaGlmZnLDqSBsZSB0b2tlbg==",
  "token_type": "Bearer",
  "not-before-policy": 0,
  "session_state": "f1fc6145-c300-4807-821f-885df6328ca5",
  "scope": "profile email"
}
```

On récupère 2 jetons, le premier étant celui identifiant l'utilisateur qui est requis pour toute interrogation d'une route de l'API (300 secondes de validité ici d'après le champ *expires_in*). Lorsque celui-ci expire, le second jeton appelé *refresh token* est utilisé pour faire une demande d'un nouveau jeton sans qu'une identification de l'utilisateur ne soit à nouveau demandée. Si le jeton *refresh token* a expiré (1 800 secondes), le processus d'authentification de l'utilisateur est à nouveau requis.

Document B7 : Extrait du courriel d'Élodie et sa pièce jointe

Bonjour,

Comme prévu, je vous fais un petit compte rendu de mes observations sur l'*API* utilisée dans l'application de gestion des formations *Sdisform*.

J'ai remarqué une chose qui me semble importante : aucune route n'est actuellement soumise à authentification de l'utilisateur qui l'appelle. Seul le code de l'application qui utilise l'*API* vérifie l'utilisateur connecté avant d'y faire appel. Ceci me semble un peu préoccupant. Qu'en pensez-vous ?

J'ai par ailleurs un petit souci sur lequel vous pourriez peut-être m'aider. J'ai testé les différentes routes avec *Postman* (application qui permet de tester les points d'entrée -*endpoint* en anglais- des *API*) mais la route pour mettre à jour une formation ne semble pas fonctionner, je récupère un message d'erreur et je ne parviens pas à en trouver la cause. Pourtant, la formation F204591 que je veux modifier en changeant son public concerné existe bien dans la base de données. Je vous joins les captures d'écran afin de visualiser cela.

Merci d'avance, cordialement.

Élodie

Extrait de la table **Formations** sous *MySQL* :

code	libelle	date	public_concerne	lieu
F123456	Le massage cardiaque	2025-01-01	Volontaires et professionnels	Ville de N.
F204591	Formation extincteurs	2024-04-01	Volontaires	Caserne de B.
F469875	Formation extincteurs	2024-01-01	Volontaires et pro	Caserne de M.

Copie d'écran de la requête *POST* vers le point d'entrée avec les données de formation passées en *JSON*.

Copie d'écran de la réponse à la requête reçue sous forme de flux *JSON* contenant l'erreur.

Document B8 : Extrait de la classe AuthentificationUtils avec les méthodes d'obtention du jeton selon le contexte (authentification ou rafraichissement)

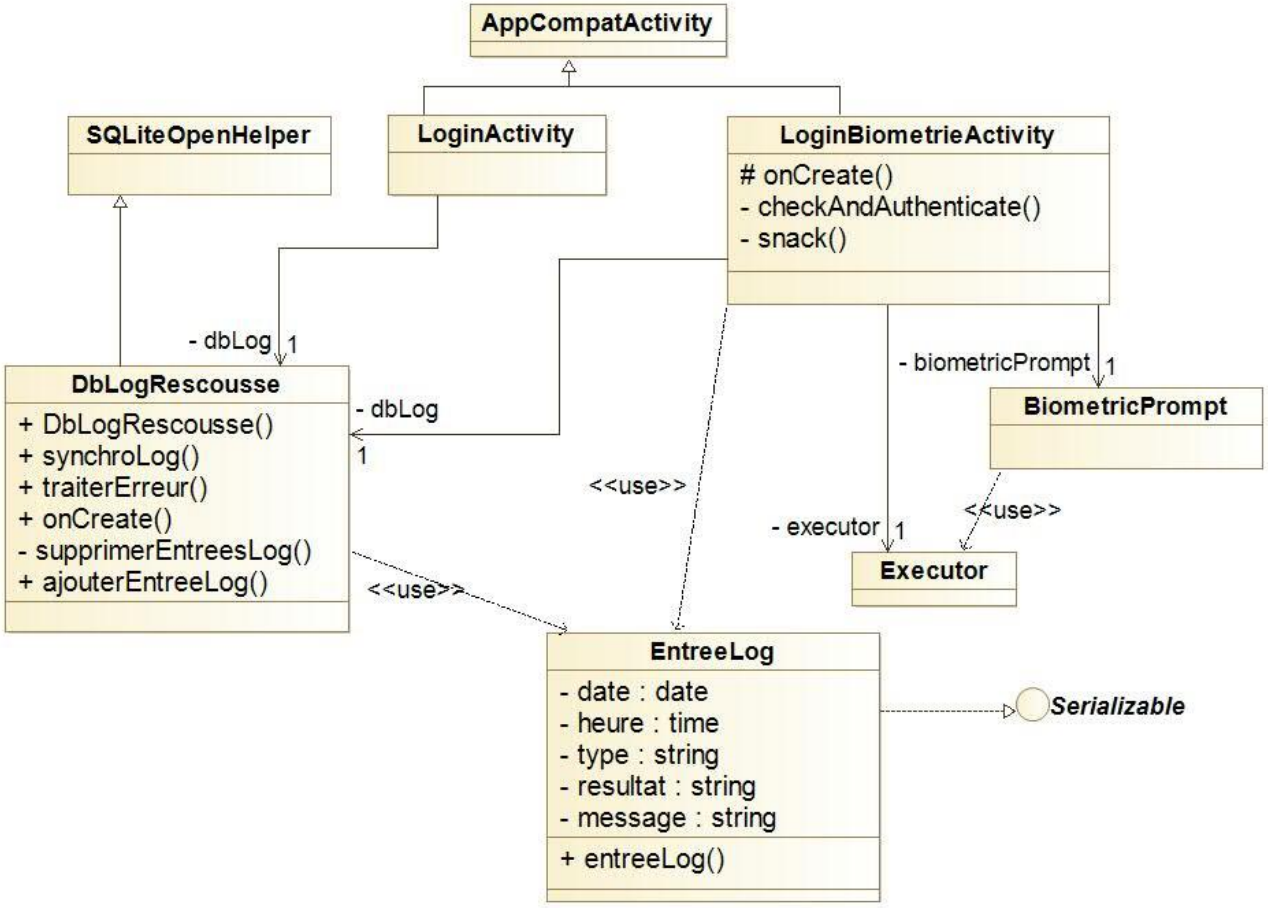
```
public class AuthentificationUtils {
  private String token;
  private String refreshToken;
…
  /*
  * Cette méthode rafraichit les jetons à partir du refreshToken actuel.
  * Elle utilise la méthode privée requestToken en passant dans une ArrayList
  * les paramètres 'refresh_token', 'client_id' et 'grant_type'
  * Le 'grant_type' valant cette fois-ci 'refresh-token'
  * @return le nouveau token
  */
  public String getToken() {
  // *****    A COMPLETER   *****

}

  /*
  * récupère les jetons à partir de l'utilisateur passé en paramètre.
  * Elle utilise la méthode privée requestToken en passant dans une ArrayList
  * les paramètres 'user_name', 'password', 'client_id' et 'grant_type'
  * @param l'utilisateur
  * @return le nouveau token
  */
  public String getToken(User leUser) {
    List<NameValuePair> params = new ArrayList<NameValuePair>();
    params.add(new BasicNameValuePair("username", leUser.getLogin()));
    params.add(new BasicNameValuePair("password", leUser.getPassword()));
    params.add(new BasicNameValuePair("client_id", "gsic_api_rolebased"));
    params.add(new BasicNameValuePair("grant_type", "password"));
    this.requestToken(params);
    return this.token;
  }
  /*
  * méthode technique assurant la connexion avec le serveur pour récupérer
  * les jetons sur le serveur SSO.
  * les jetons récupérés sont affectés aux attributs de la classe correspondants
  * @param ArrayList les paramètres nécessaires à la requête
  */
  private void requestToken(List<NameValuePair> params){
    String postURL = "https://sso.gsic.fr:8443/realms/gsic/protocol/openid-connect/token";
    try (CloseableHttpClient httpClient = HttpClients.createDefault()) {
      HttpPost post = new HttpPost(postURL);
      UrlEncodedFormEntity ent = new UrlEncodedFormEntity(params, "UTF-8");
      post.setEntity(ent);
      HttpResponse responsePOST = httpClient.execute(post);
      HttpEntity entity = responsePOST.getEntity();
      String getResponseString = EntityUtils.toString(entity);
      JSONObject leJSON = new JSONObject(getResponseString);
      this.token = leJSON.getString("access_token");
      this.refreshToken = leJSON.getString("refresh_token");
    } catch (Exception e) {
        e.printStackTrace();
        …
    }
  }
}
```

Documents associés au dossier C

Document C1 : Diagramme de classes partiel de l'application Rescousse



Document C2 : Extrait des classes DbLogRescousse et EntreeLog

```
public class EntreeLog implements Serializable {

    private LocalDate date = LocalDate.now();
    private LocalTime heure = LocalTime.now();
    private String type;
    private String resultat;
    private String message;

    /** Chaque attribut de la classe possède un accesseur en lecture :
getDate(), getHeure(), getType(), getResultat() et getMessage() **/

    public EntreeLog(String pType, String pResultat, String  pMessage) {    }
}
```

```
public class DbLogRescousse extends SQLiteOpenHelper {
    protected static final String DATABASE_NAME = "DbLogRescousse.db";
    private static final int DATABASE_VERSION = 1;
    private static final String TABLE_NAME = "Log";

    public DbLogRescousse(@Nullable Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
    /**
     * La méthode synchroLog transmet l'ensemble des logs de la base locale
     *       SQLite DbLogRescousse.db au serveur.
     * Ces logs seront insérés dans la base de données du serveur,
     * puis ils sont supprimés de la base locale SQLite
     */
    public void synchroLog() { … }

    public void traiterErreur(String msg, Exception ex) { … }

    /** Crée la table Log, id est la clé primaire en autoincrément
     * @param db Base de données SQLite dans laquelle créer la table Log */
    @Override
    public void onCreate(SQLiteDatabase db) {
        try {
            String strReq = "CREATE TABLE " + TABLE_NAME
                    + " (id INTEGER PRIMARY KEY AUTOINCREMENT, "
                    + "date TEXT  NOT NULL, "
                    + "heure TEXT NOT NULL, "
                    + "type TEXT NOT NULL, "
                    + "resultat TEXT NOT NULL, "
                    + "message TEXT NOT NULL );";
            db.execSQL(strReq);
        } catch (Exception ex) {
            traiterErreur("création table", ex);
        }
    }

    /** Utilisée par synchroLog afin de supprimer tous les enregistrements
           de la table Log de la base de données SQLite */
    private void supprimerEntreesLog() {
        try {
          //Ouvre la base de données dans laquelle lire ou insérer les données
          //Si celle-ci n'existe pas la méthode onCreate est alors appelée
            SQLiteDatabase db = this.getWritableDatabase();
            String sql = "DELETE FROM " + TABLE_NAME + ";";
            db.execSQL(sql);
        } catch (Exception ex) {
            traiterErreur("suppression log", ex);
        }
    }

    /** Insére les données de l'entrée log dans la table Log */
    public void ajouterEntreeLog(EntreeLog uneEntreeLog) {
                //   *****    A COMPLETER   *****
    }
}
```

Document C3 : Extraits de la documentation SQLite

Extrait : Types de données SQLite	
TEXT : chaîne de caractères	**INTEGER** : entier signé
REAL : nombre réel en virgule flottante	**NULL** : valeur NULL
SQLite ne fournit pas de types spécifiques pour Date et Heure. Le type TEXT peut être utilisé à la place.	
Extrait : Insertion de table	
INSERT INTO table_name (column1_name [, column2_name, ...]) VALUES (value_column1[, value_column2_name, ...]);	

Document C4 : Extrait des recommandations de la CNIL au sujet de la journalisation

La Commission recommande que les opérations de création, consultation, modification et suppression des données à caractère personnel et des informations contenues dans les traitements auxquels la journalisation est appliquée fassent l'objet d'un enregistrement comprenant l'auteur individuellement identifié, l'horodatage, l'équipement utilisé ainsi que la nature de l'opération réalisée. Il convient notamment d'éviter de dupliquer au sein des journaux les données concernées par le traitement. Cette journalisation peut être intégrée au niveau applicatif ou bien gérée au niveau technique au moyen des ressources logicielles utilisées par l'application.

Document C5 : Extrait de la classe LoginBiometrieActivity

```
public class LoginBiometrieActivity extends AppCompatActivity {

   private Executor executor;
   private BiometricPrompt biometricPrompt;
   private BiometricPrompt.PromptInfo promptInfo;
   private DbLogRescousse dbLog;


   /** Vérifie le bon fonctionnement du système de biométrie, puis fait appel
    * à la méthode d'authentification de l'objet biometricPrompt. */
   private void checkAndAuthenticate() { }


   /** Affiche un message pour l'utilisateur – ancienne méthode toast */
    private void snack(String text)  {  }
```

Suite de la classe page suivante

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login_biometrie);
        dbLog = new DbLogRescousse(getApplicationContext());
        Intent rescousseActivite = new Intent(this, RescousseActivity.class);

        executor = ContextCompat.getMainExecutor(this);

        /* Initialisation de l'objet biometricPrompt qui permet de gérer la
                                    communication avec le système de biométrie */
        biometricPrompt = new BiometricPrompt
        (
            LoginBiometrieActivity.this, executor,
            new BiometricPrompt.AuthenticationCallback()
            {
                @Override
                public void onAuthenticationError(int errorCode,
                                                  @NonNull CharSequence errString)
                {
                    super.onAuthenticationError(errorCode, errString);
                    // cas LoginBiometrie en erreur :
                    // enregistrement du log et
                    // affichage du message d'erreur à l'utilisateur
                    //   *****    A COMPLETER   *****
                }
                @Override
                public void onAuthenticationSucceeded(@NonNull
                                    BiometricPrompt.AuthenticationResult result)
                {
                    super.onAuthenticationSucceeded(result);
                    // cas LoginBiometrie en succès : enregistrement du log
                    //   *****    A COMPLETER   *****

                     // démarrer activity suivante RescousseActivity
                     startActivity(rescousseActivite);
                }
            }
        );

        promptInfo = new BiometricPrompt.PromptInfo.Builder()
                    .setTitle("Authentification biométrique Rescousse")
                    .setSubtitle("Se connecter avec son identifiant biométrique")
                    .setAllowedAuthenticators(BIOMETRIC_STRONG | DEVICE_CREDENTIAL)
                    .build();

        checkAndAuthenticate();
    }
}
```