IPST-CNAM
Intranet et Designs patterns
NSY 102
Lundi 7 Avril 2025

Durée : **2 h 45**
Enseignants : LAFORGUE Jacques

1ère Session NSY 102

1ère PARTIE – SANS DOCUMENT (durée: 1h15)

1. QCM (35 points)

Mode d'emploi :

Ce sujet est un QCM dont les questions sont de 3 natures :

- **les questions à 2 propositions**: dans ce cas une seule des 2 propositions est bonne.
  - +1 pour la réponse bonne
  - –1 pour la réponse fausse
- **les questions à 3 propositions** dont 1 seule proposition est bonne
  - + 1 pour la réponse bonne
  - –½ pour chaque réponse fausse
- **les questions à 3 propositions** dont 1 seule proposition est fausse
  - + ½ pour chaque réponse bonne
  - –1 pour la réponse fausse

Il s'agit de faire une croix dans les cases de droite en face des propositions.

On peut remarquer que cocher toutes les propositions d'une question revient à ne rien cocher du tout (égal à 0).

Si vous devez raturer une croix, faites-le correctement afin qu'il n'y ait aucune ambiguïté.

N'oubliez pas d'inscrire en en-tête du QCM, votre nom et prénom.

Vous avez droit à **4 points** négatifs sans pénalité.

NOM:	PRENOM:

Dans la démarche d'architecture d'un SI, l' **Architecture Applicative** réunit :		Q 1.
1	la **Configuration Architecturale**, l' **Architecture Technique** et l' **Architecture Physique.**	
2	l' **Architecture Fonctionnelle** et l' **Architecture du Système**	
3	la **Configuration Architecturale** et la **Dynamique de l' Architecture**	

La **Configuration Architecturale** est la décomposition d'un Système d'Information en composants logiciels et sous-composants.		Q 2.
1	OUI	
2	NON	

Une architecture logicielle est la représentation d'un programme sous la forme d'un diagramme de classe UML qui décrit les relations entre les composants (les classes) du programme		Q 3.
1	OUI	
2	NON	

Dans la démarche de conception d'un SI, la mise en évidence des Designs Patterns se fait :		Q 4.
1	avant la création du Diagramme de Communication	
2	pendant la création du Diagramme de Communication	
3	après la création du Diagramme de Communication (lors de la création des Diagrammes de Classe)	

Dans la démarche de conception d'un SI, un CONNECTEUR est toujours orienté dans le sens de l'appelant vers l'appelé		Q 5.
1	OUI	
2	NON	

Dans la description de l'architecture technique, un CONNECTEUR est un lien de dépendance entre deux composants qui peut être réalisé par le principe du design pattern de l'injection de dépendance.		Q 6.
1	OUI	
2	NON	

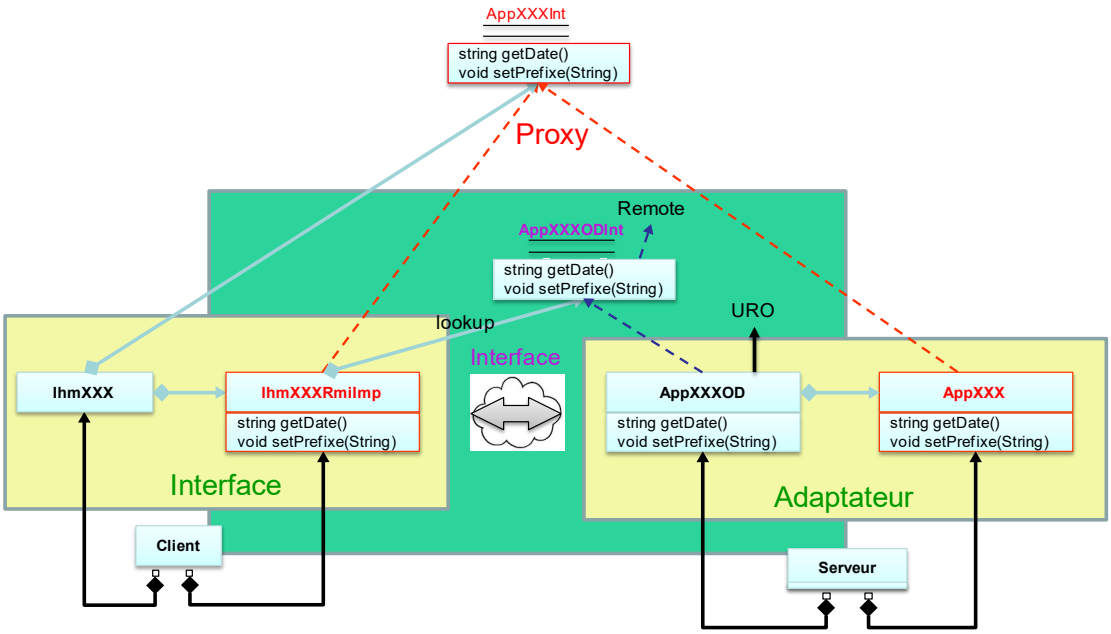
Dans un Diagramme de Communication, deux sous-composants appartenant à 2 composants logiciels distants différents peuvent directement communiquer entre eux.		Q 7.
1	OUI	
2	NON	

Dans la démarche de conception logicielle utilisée dans ce cours, le diagramme de communication décrit des liens de communication entre chacun des composants et sous-composants. Ces liens sont :		Q 8.
1	toujours des communications distantes.	
2	des liens non orientés. Leurs orientations sera résolues à l'étape suivante des diagrammes de classes.	
3	Des liens orientés, distants entre les composants logiciels, ou locaux entre les sous-composants.	

Dans une architecture à base de composant, le conteneur assure la localisation et la résolution des dépendances entre les composants.		Q 9.
1	OUI	
2	NON	

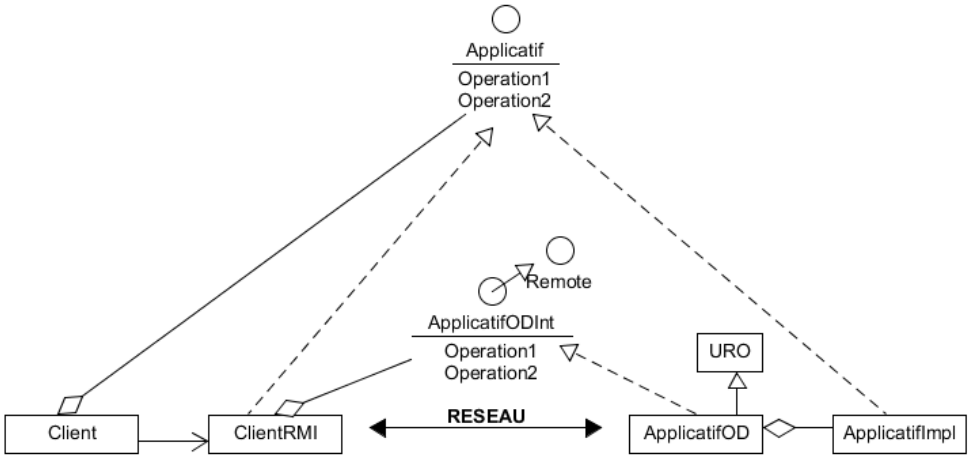
Soit un objet quelconque Obj (instance de la classe A qui implémente l'interface IntA). En Java RMI, il est très facile de transformer cet objet en un objet distant. Pour cela il suffit de :		Q 10.
1	créer une Adaptateur de la classe A. Cet adaptateur hérite de UnicastRemoteObject et implémente l'interface distante AdaptIntA qui hérite de Remote.	
2	créer un proxy de A . Ce proxy hérite de UnicastRemoteObject et implémente l'interface de A.	

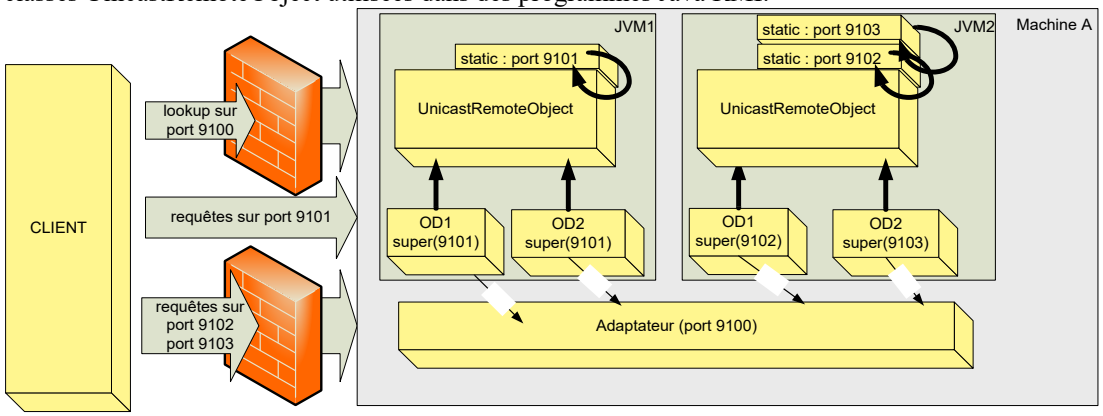
Ceci est le diagramme de classe d'un système composé d'un client IHM (classe IhmXXX) et de son applicatif (AppXXX) que l'on veut rendre distant. Q 11.



Le rôle de IhmXXXRmiImp est de :

1	implémenter les méthodes distantes décrite par l'interface RMI AppXXODInt.	
2	implémenter les méthodes de l'interface AppXXXInt qui réalisent l'appel des méthodes distantes implémentées par AppXXXOD.	

Ceci est le diagramme de classe d'un système composé d'un client et de son applicatif que l'on veut rendre distant. 		Q 12.
1	ClientRmi est un proxy de communication de ApplicatifOD.	
2	ApplicatifOD est un adaptateur à l'interface ApplicatifODInt pour ApplicatifImpl.	
3	ClientRmi est un proxy de ApplicatifImpl.	

Soit le schéma suivant qui représente un fonctionnement possible de plusieurs serveurs de socket des classes UnicastRemoteObject utilisées dans des programmes Java RMI. 		Q 13.
1	On peut créer un nouvel OD dans la JVM1 qui s'exécute sur le port 9102.	
2	Sur la machine A, on peut créer une nouvelle JVM3 dans laquelle, on crée un nouvel OD qui s'exécute sur le port 9103.	
3	Dans la JVM2, on peut créée un nouvel objet distant RMI sur le port 9102.	

Avec l'API RMI de Java, un client distant d'un serveur peut utiliser les méthodes d'un Objet Distant instancié sur le serveur, en utilisant un Proxy de Communication instancié sur le client. Ce proxy de communication est :		Q 14.
1	Un objet instancié par la méthode prédéfinie **toStub.**	
2	Un objet instancié par la méthode **lookup.**	
3	Un objet instancié par la méthode **rebind.**	

Soit un objet quelconque Obj (instance de la classe A qui n'hérite pas d'une autre classe). En Java RMI, il est très facile de transformer cet objet en un objet distant. Pour cela il suffit de :		Q 15.
1	faire que la classe A implémente l'interface Remote.	
2	faire que la classe A implémente l'interface Serializable, puis écrire cet objet dans un annuaire RMI.	
3	faire hériter A de UnicastRemoteObject, mettre les méthodes dans une interface publique (I) qui hérite de Remote, et faire que la classe A implémente l'interface I.	

En RMI, le Proxy de Communication d'un Objet Distant est un objet sérialisé qui peut être envoyé à n'importe quel client Java afin que celui-ci peut l'utiliser pour appeler les méthodes distantes de l'Objet Distant.		Q 16.
1	OUI	
2	NON	

Un Design Pattern (DP) est :		Q 17.
1	une norme de description des interfaces entre les composants d'une architecture logicielle orientée objet.	
2	un template de code représentant l'implémentation d'un concept informatique qui répond à une problématique récurrente dans la réalisation d'un Système d'information.	
3	une description spécifique d'un principe général de conception décrit sous la forme d'un diagramme de classe	

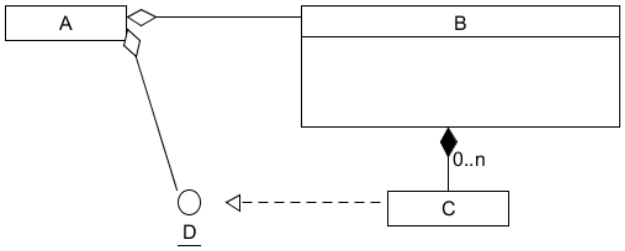
Dans un DP Singleton toutes les méthodes d'accès (getteur et setteur) à l'objet unique sont des méthodes statiques.		Q 18.
1	OUI	
2	NON	

Il est interdit de créer des setteurs sur un singleton permettant de modifier les attributs du singleton.		Q 19.
1	OUI	
2	NON	

Dans le DP Factory, les produits du Factory sont des objets uniques. Ces objets sont donc vus par le reste du programme comme des singletons.		Q 20.
1	OUI	
2	NON	

Dans le DP Factory, les produits du Factory sont nécessairement stockés en mémoire du Factory.		Q 21.
1	OUI	
2	NON	

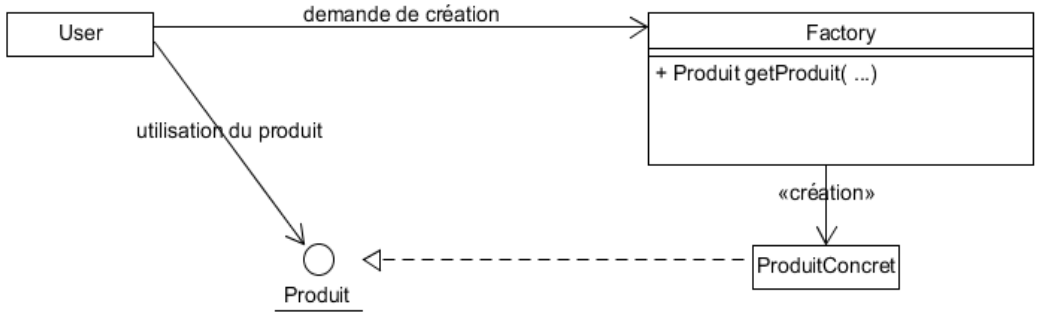
Ce DP est celui du Factory. Q 22.



La signification des lettres A, B, C et D est :

1	A = Client; B=Product (interface); C=Concrete Product; D = Factory	
2	A=Client; B=Factory; C=Concrete Product; D=Product (interface);	
3	A=Factory; B = Concrete Product; C=Product (Interface); D=Client	

Q 23.



Le rôle de cette variante du DP Factory permet de stocker les produits créés en mémoire du factory.

1	OUI	
1	NON	

Quelques soient les variantes du DP Factory, le Client utilise toujours les produits créés par le factory à travers une interface ou une classe abstraite.		Q 24.
1	OUI	
2	NON	

Dans le DP Observateur, la communication entre l'Observer (consommateur d'évènement) et l'Observable (producteur d'évènement) est nécessairement asynchrone car la communication se fait toujours par l'envoi d'un message sans valeur de retour.		Q 25.
1	OUI	
2	NON	

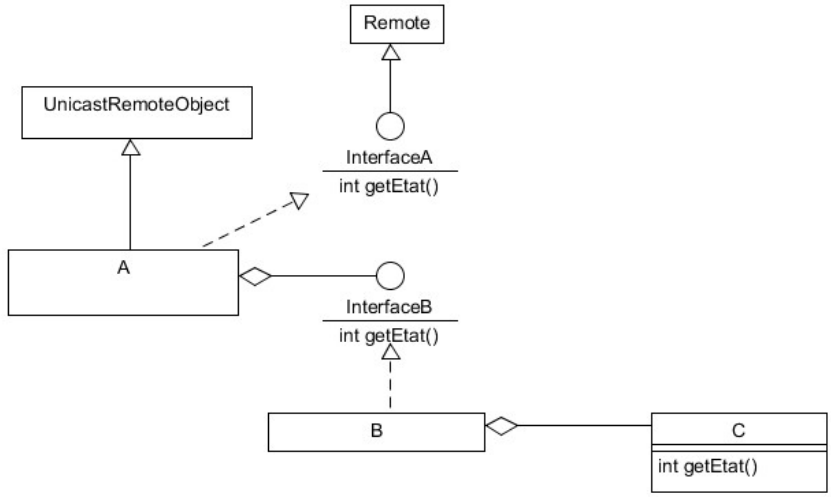
Dans le DP MVC (Model Vue Controleur) l'Observable (objet observé) est dans le Model.		Q 26.
1	OUI	
2	NON	

Dans le DP Observateur, le mode de communication entre l'objet observateur et l'objet observé peut être :		Q 27.
1	Pull synchrone	
2	Push synchrone	
3	Push asynchrone	

Dans DP Décorateur, le Composant Concret et toutes les classes de décorations héritent d'une même classe abstraite.		Q 28.
1	OUI	
2	NON	

Les descriptions suivantes sont des modèles de communication asynchrones :		Q 29.
1	un serveur pousse ses évènements dans une file d'attente par client connecté (intermédiaire), et les clients tirent ses évènements à leur rythme	
2	un serveur appelle la méthode distante d'un client afin de lui pousser l'évènement	

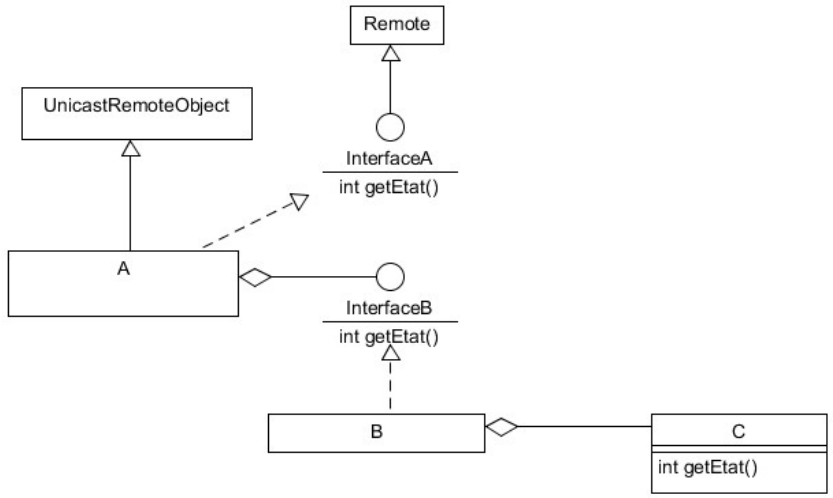
Q 30.



Ce diagramme de classe est la conception d'un Objet Distant suivant le modèle :		
1	d'un DP Proxy	
2	d'un DP Adaptateur	

Dans la communication asynchrone via un "canal d'évènement" entre un producteur et un consommateur :		Q 31.
1	le consommateur utilise un proxy de producteur (et non le producteur directement), afin de lui pousser un évènement	
2	le producteur utilise un proxy de consommateur (et non les consommateurs directement), afin de lui pousser un évènement	

Le principe général d'un MOM (Message Oriented Middleware) est que tous les composants connectés, à un même canal d'évènement du bus du MOM, en mode Queue, reçoivent tous les évènements publiés dans le fournisseur de service MOM (Provider)		Q 32.
1	OUI	
2	NON	



Ce diagramme de classe est la conception d'un Objet distant dans lequel :		Q 33.
1	A est un Adaptateur de B	
2	A est un Adaptateur de C	

Un canal d'évènement est constitué de deux DP : un DP Factory permettant de créer des évènements et un DP Iterator permettant de parcourir ces évènements.		Q 34.
1	OUI	
2	NON	

Laquelle des descriptions suivantes est un principe de communication synchrone :		Q 35.
1	le producteur dépose à son rythme ses évènements dans une file. Le ou les consommateurs peuvent alors récupérer ces évènements	
2	le producteur pousse ("push") chaque évènement vers chacun des consommateurs via une méthode distante qui retourne un état de consommation	

Fin du QCM

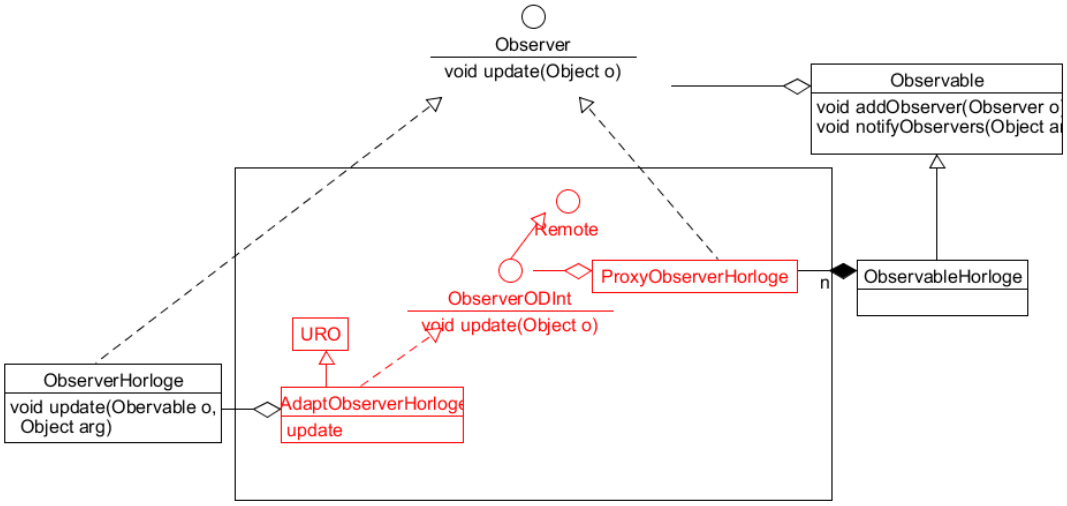
Suite (Tournez la page)

2. Questions libres (15 points)

Chaque question est notée sur 5 points.
*Vous répondez à ces questions sur une **copie vierge double** en mettant bien le numéro de la question, sans oublier votre nom et prénom.*
Vous mettez le QCM dans la copie vierge double.

QUESTION NUMERO 1

Le diagramme suivant représente le DP Observateur Distant :



Expliquez, avec précision, le rôle et le fonctionnement de ce DP.
Quel est le principe de conception informatique de la partie encadrée en rouge ?

QUESTION NUMERO 2

Nous avons vu en cours 5 modèles de conception d'un Objet Distant dont celle de la conception d'un OD par double adaptateur et celle par proxy.
Écrivez chacun des diagrammes de classe de ces 2 modèles.

QUESTION NUMERO 3

Ecrivez le diagramme UML du DP de **l'injection de dépendance** et expliquez le rôle de ce DP.

Fin de la 1ère partie sans document

2ème PARTIE – AVEC DOCUMENT (durée: 1h30)

3. PROBLEME [50 points]

On se propose de faire un Système d'Information (SI) de surveillance d'une entreprise à fort risque industriel pour la sécurité de ses employés et de l'entreprise, avec des caméras « intelligentes ».

Les caméras sont des caméras IP diffusant un flux vidéo (voir annexe).
Le flux vidéo de chaque caméra est récupéré par le Serveur Principal (COMPOSANT 1) qui réalise 3 tâches, en parallèle et en continu :
- réaliser une chaîne de traitements d'IA sur N images consécutives.
- enregistrer toutes les images du flux vidéo sous la forme d'un fichier JPG par tranche de 24h.
- envoyer toutes les images, en temps réel, vers des Postes de Supervision (COMPOSANT 2) (nombre indéterminée).

Au lancement, le Serveur Principal se configure avec la liste des adresses IP de toutes les caméras.
La chaîne de traitement est créée par une injection de dépendance à partir d'un fichier de configuration.

Pour des raisons de performance, chacun des traitements IA sont effectivement réalisés par un autre composant : le Serveur de traitement IA (COMPOSANT 3). Ce composant contient la liste des traitements qu'il est possible d'enchaîner.
Si un de ces traitements détecte une anomalie, il notifie des alertes dédiées (identification de la caméra, datage, nature de l'anomalie,…) au Serveur Principal qui, à son tour, dispatche ces alertes à des Postes de Supervision (reconnaissance faciale de personne inconnue, présence anormale dans des zones prédéterminées, agitation, personne en danger, tenue de sécurité non adaptée, …etc...).
L'exécution d'une chaîne de traitement peut donc notifier plusieurs alertes.

Chaque Poste de Supervision est dédié à un groupe de caméra déterminé par un fichier de configuration, qui reçoit les alertes et les images le concernant.
Un Poste de Supervision affichent ses alertes et les images en continu de ses caméras dans autant de zones graphiques. Chaque zone d'une caméra est mise en évidence concernant l'alerte.
Un opérateur peut alors sélectionner une des caméras pour visualiser (retour arrière, avance), dans une zone graphique spécifique, le fichier d'enregistrement concerné, situé sur le Serveur Principal.

Une IHM Principale (COMPOSANT 4) permet de configurer à distance les Postes de Supervision afin de répartir les caméras en fonction des Postes de Supervision disponibles. Cette IHM centralise aussi l'affichage de toutes les alertes.
Cette IHM permet aussi de visualiser n'importe quel fichier JPG stocké sur le disque dur du Serveur Principal.
Cette IHM permet aussi de créer le fichier de configuration de la chaîne de traitements d'IA.

1/ **[15 points]**
Faites le diagramme de communication de ce Système d'Information. Il y a donc 4 composants logiciels à décrire qui communiquent entre eux de manière distant.
Commentez votre schéma (rôles des composants et sous-composants, comportement dynamique général, échanges des informations, localisation des données).
Nous rappelons que ce schéma doit permettre de connaître vos choix d'organisation des composants, le sens des communications, et les designs patterns envisagés.

2/ **[35 points]**
Faites le(s) diagramme(s)e de classe UML des [COMPOSANT 1] et [COMPOSANT 2] en mettant en évidence les Designs Patterns utilisés.
Commentez chacun de(s) diagramme(s).

Précisions :
Un composant applicatif [COMPOSANT X] correspond à une JVM ou process. Cela signifie que les COMPOSANTS X communiquent sur le réseau à travers des interfaces distantes.
Ainsi, pour une description précise de vos diagrammes de classe, on fait le choix que toutes les communications distantes entre les composants sont réalisées en RMI (utilisation de la classe URO = UnicastRemoteObject et de l'interface Remote).

Annexe : Flux vidéo en RTSP (Real-Time Streaming Protocol). Le serveur utilise l'API OpenCV pour récupérer les images de chaque caméra sous la forme d'un objet de classe **Mat.**
Exemple de code mettant en évidence que **Mat** (abréviation de Matrix) est la structure de données principale d'OpenCV. Il représente une image sous forme de matrice de pixels.

```
 String rtspUrl = "rtsp://username:password@IP_CAMERA:554/stream";  // L'URL de la
caméra
VideoCapture capture = new VideoCapture(rtspUrl);
Mat frame = new Mat();
VideoWriter writer = new VideoWriter(……) ;
while (capture.read(frame)) { // Récupère une image
  if (!frame.empty()) {
    (exemple 1 ) Imgcodecs.imwrite("frame_" + frameCount + ".jpg", frame); //
Sauvegarde l'image
    (exemple 2) envoyer frame à une autre classe
    (exemple 3) writer.write(frame); // Ajoute l'image à la vidéo pendant 24h
   }
}
```

Fin du sujet