

24/02/2025

Wroblewksi Martin

**CER**

Rôles

* Tuteur : Mohamed
* Animateur : Martin.A
* Secrétaire : Louis
* Scribe : Martin.W
* Gestionnaire : Ludovic

Analyse du besoin

**Analyse** **du** **besoin** :

Nous cherchons à connaitre la conteneurisation des applications .NET.

**Généralisation** :

Je résous le problème grâce à l’hébergement et transmission de données en temps réel.

Sommaire

**Sommaire**

[I. Mise en situation 2](#_Toc191278814)

[A. Définition des mots clés. 2](#_Toc191278815)

[B. Contexte. 3](#_Toc191278816)

[C. Contraintes. 3](#_Toc191278817)

[D. Livrable. 3](#_Toc191278818)

[E. Problématique 3](#_Toc191278819)

[II. Résolution. 4](#_Toc191278820)

[A. Plan d’action. 4](#_Toc191278821)

[B. Résolution. 5](#_Toc191278822)

[III. Cours 7](#_Toc191278823)

[A. Traitement séquentiel et parallèle 7](#_Toc191278824)

[B. Mutex 8](#_Toc191278825)

[C. Sémaphore 8](#_Toc191278826)

CER

# Mise en situation

## Définition des mots clés.

Mots clés :

* DMZ : Un réseau DMZ (réseau de zone démilitarisé) est un réseau périmétrique qui protège le LAN interne d'une organisation contre le trafic non fiable.
* Docker/Conteneur : Un conteneur Docker est un environnement d'exécution contenant tous les composants nécessaires, tels que le code, les dépendances et les bibliothèques, nécessaires pour exécuter le code de l'application sans utiliser les dépendances de la machine hôte.
* Mécanisme : Un mécanisme désigne un ensemble de processus ou de méthodes permettant d'accomplir une tâche spécifique. En informatique, cela peut faire référence à des protocoles, des algorithmes ou des structures mises en place pour exécuter une fonction précise.
* État d’avancement : Il s'agit du niveau d'évolution d'un projet, d'un processus ou d'une tâche à un instant donné. Il peut être représenté par des indicateurs comme un pourcentage d'achèvement, des jalons atteints ou des phases complétées.
* Unité de traitement : En informatique, une unité de traitement désigne un composant chargé d'exécuter des instructions, comme un processeur (CPU) ou une unité de traitement graphique (GPU). Elle joue un rôle clé dans l'exécution des calculs et des opérations logiques.
* Gestion des ports IP : Cela fait référence à l'administration et à la configuration des ports réseau utilisés par un système informatique pour communiquer via le protocole IP. Elle permet d'assurer la sécurité et la bonne répartition du trafic réseau entre les services.
* DotNet (.NET) : .NET est un framework de développement créé par Microsoft, utilisé pour développer, exécuter et déployer des applications sur différentes plateformes (Windows, Linux, macOS). Il supporte plusieurs langages de programmation comme C# et F#, et permet de créer des applications web, mobiles, cloud et desktop.
* Enregistrement de données : L'enregistrement de données désigne le processus de stockage et de conservation des informations dans une base de données, un fichier ou un autre support numérique. Il peut inclure des aspects tels que la structuration, la validation et la sécurisation des données.
* Outils de simulation d’avancement : Ce sont des logiciels ou des applications permettant de modéliser et d'analyser l'évolution d'un projet ou d'un système en fonction de paramètres définis. Ils sont souvent utilisés pour prévoir les résultats futurs et optimiser la gestion des ressources.
* Hébergement d’application : Cela désigne le processus de mise en ligne et de gestion d'une application sur un serveur ou une infrastructure cloud. L'hébergement permet aux utilisateurs d'accéder à l'application via Internet ou un réseau local. Les solutions d'hébergement peuvent être partagées, dédiées ou basées sur le cloud.

## Contexte.

Franck doit transmettre en temps réel via un mécanisme d’envoie concernant l’état d’avancement de la fabrication à son client, tout en hébergeant son application dans un conteneur Docker sous .NET.

## Contraintes.

Les contraintes sont :

* Code
* Mécanisme d’envoi (distance)
* Problèmes liés au nouvel environnement
* Transmission en temps réel

## Livrable.

Code en C#, conteneur en DotNet (Docker).

## Problématique

Comment assurer cette transmission en respectant les contraintes d’hébergement et de gestion des données ?

# Résolution.

## Plan d’action.

Le plan d’action se déroule en plusieurs étapes :

1. Sélectionner la technologie de transmission de données
2. Définir les composants du sytème
3. BDD
4. Configurer Docker + DMZ + CI/CD + Sécurité (SSL, …)
5. Code (Back+Front) + implémenter les Web Sockets
6. Solution de résilience avec un CRC
7. Test de charge

CER

## Résolution.

**Sélection de la Technologie de Transmission de Données**

Pour transmettre l'état d'avancement de la fabrication en temps réel, Franck peut utiliser :

* WebSockets : Idéal pour une communication bidirectionnelle temps réel entre le serveur et le client.
* MQTT : Adapté aux environnements IoT et faible consommation.
* SignalR (DotNet) : Permet une transmission efficace et intégration facile avec .NET.
* API REST avec polling (moins efficace pour du temps réel).

Franck optera pour WebSockets ou SignalR afin d'assurer une mise à jour instantanée de l'état d'avancement.

**Définition des Composants du Système**

Le système sera composé des éléments suivants :

* Application de simulation (existant, à interfacer avec la solution de transmission).
* Serveur d'application DotNet (gestion de la communication en temps réel et API REST).
* Client Web (dashboard interactif pour visualiser l'état d'avancement).
* Base de données pour stocker l'historique de production.
* Docker pour déployer l'application dans la DMZ.

**Base de Données (BDD)**

Franck doit définir un schéma de BDD pour stocker les états de production :

* Table "production" : identifiant, timestamp, état actuel, unité de traitement.
* Table "alertes" : anomalies ou retards de production.
* Table "utilisateurs" : gestion des accès.

**Configuration de Docker, DMZ, CI/CD et Sécurité**

* Docker : Création d'un conteneur DotNet pour exécuter le serveur d'application.
* DMZ : Validation des règles de firewall et exposition sécurisée de l'application.
* CI/CD : Automatisation des déploiements avec GitHub Actions ou Jenkins.
* SSL/TLS : Mise en place de certificats pour sécuriser la communication.
* Gestion des ports : Redirection des flux WebSockets et REST via le proxy de la DMZ.

# Cours

## Sockets et architecture Client-Serveur

Les sockets sont une technologie de communication qui permet à deux applications de s’échanger des données via un réseau. Elles sont la base de nombreux protocoles tels que HTTP, FTP, et même WebSockets.

L’architecture client-serveur repose sur un modèle où un serveur fournit un service auquel les clients se connectent.

**Définition des Sockets**

Un socket est un point de terminaison pour la communication entre deux machines. Il repose sur des protocoles de transport comme :

* TCP (Transmission Control Protocol) → fiable, orienté connexion.
* UDP (User Datagram Protocol) → non fiable, sans connexion.

**Cycle de vie d’un socket**

1. Création du socket (client et serveur).
2. Bind (côté serveur) : association du socket à une adresse IP et un port.
3. Écoute (serveur) : attente des connexions clients.
4. Connexion (client) : demande de connexion au serveur.
5. Échange de données (lecture/écriture sur le socket).
6. Fermeture (libération des ressources).

**Principe de Fonctionnement de l'Architecture Client-Serveur**

L’architecture client-serveur est un modèle de communication où :

* Un serveur fournit des services (ex. base de données, fichiers).
* Un client envoie des requêtes au serveur pour obtenir ou envoyer des informations.

**Fonctionnement d’un système Client-Serveur**

1. Le client se connecte au serveur via un socket.
2. Le serveur écoute les requêtes et répond.
3. Le client envoie des données (exemple : message).
4. Le serveur traite la demande et renvoie la réponse.
5. Fermeture de la connexion.

Exemple : Une application de chat en temps réel utilise ce modèle, où le serveur relaye les messages entre clients.

**Transmission de Flux**

La transmission de données via un réseau repose sur deux principaux modes :

* Mode orienté connexion (TCP)
* Mode sans connexion (UDP)

**Transmission TCP (Fiable)**

**Schéma de fonctionnement**

1. Établissement de la connexion.
2. Envoi des données (découpées en paquets).
3. Réception et réassemblage côté client.
4. Confirmation de la réception.

**Avantages/Inconvénients :**

✅ Fiabilité (pas de perte de paquets).  
✅ Garantie de l'ordre des messages.  
❌ Plus lent que UDP (handshake, retransmissions).

**Transmission UDP (Rapide, mais moins fiable)**

**Schéma de fonctionnement**

1. Envoi direct des paquets sans connexion préalable.
2. Réception des paquets par le client.
3. Pas de garantie d’ordre ou de réception.

**Avantages/Inconvénients**

✅ Rapide, faible latence.  
❌ Aucune garantie d’arrivée des paquets.

Exemple : UDP est utilisé dans les jeux en ligne ou la diffusion vidéo en direct.