Temps de réponse

Contexte

On veut développer un système permettant de vérifier que des machines distantes, installées sur différents réseaux, sont toujours allumées.

L'ensemble des développements doivent être réalisés en .NET 7

Application d'administration

Dans un premier temps, une application WPF doit permettre de gérer une liste d'ordinateurs avec les informations suivantes :

- Numéro de la machine (entier, unique)
- Nom de la machine
- Description de la machine
- Date de mise en service

Il doit être possible d'ajouter, de supprimer et de modifier. L'application WPF ne doit pas accéder directement à la BDD. Elle lit et enregistre les données par l'intermédiaire d'une Web API C#.

L'application WPF doit ensuite permettre de consulter les informations de monitoring (décrites ci-dessous) pour chacune des machines, ordonnées par date du plus récent au plus ancien.

Application de monitoring

Vous devez réalisez une application console qui sera installée sur les machines à monitorer. Elle envoie à intervalle régulier à l'API un signal de présence. Cette application doit envoyer :

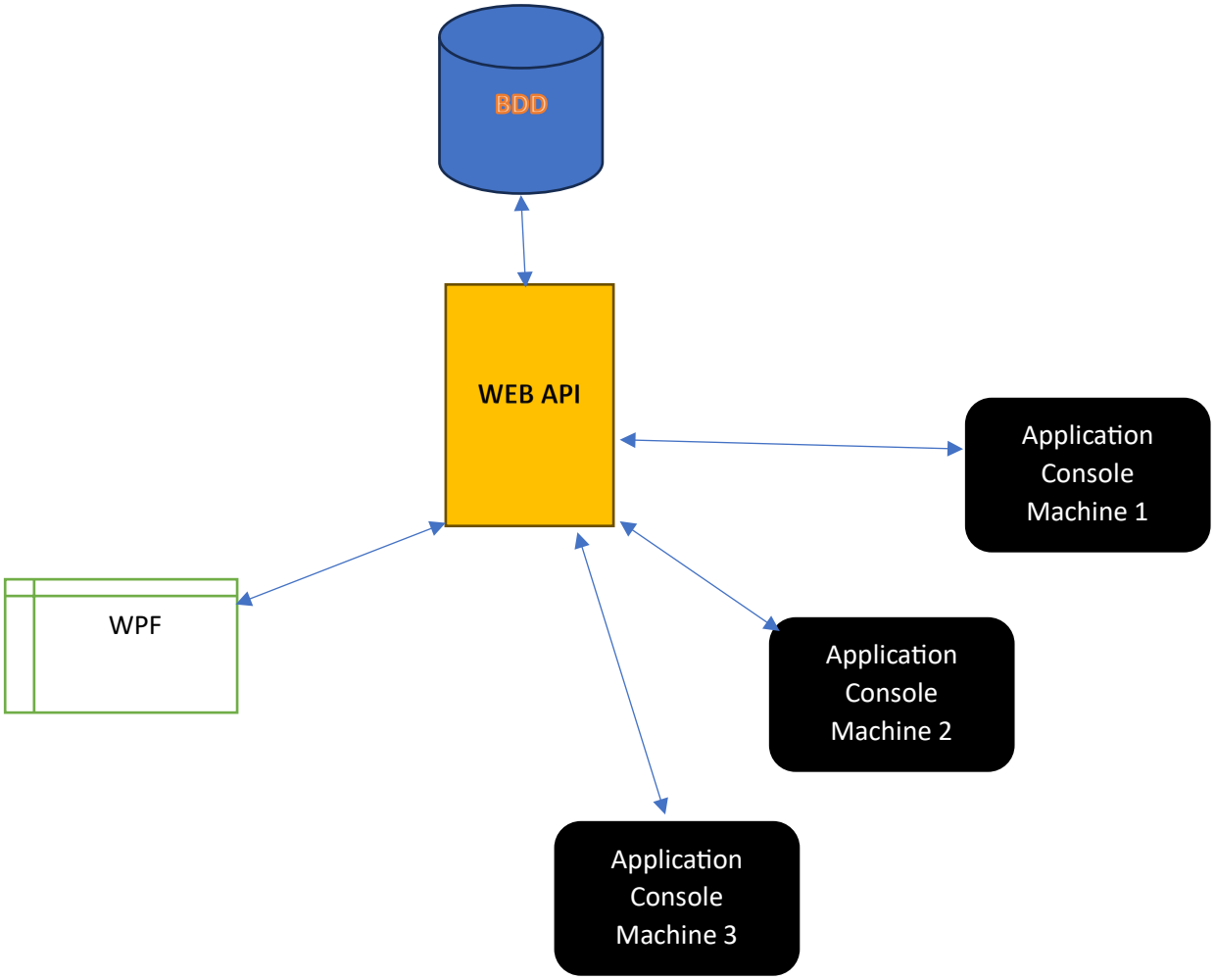
- Le numéro de la machine
- L'heure courante sur la machine

L'API enregistrera dans la BDD ces deux informations et son heure courante (pour voir si les machines distantes sont à l'heure).

Si le réseau n'est pas disponible et que l'API ne répond pas, l'application ne doit pas s'arrêter, mais simplement continuer à essayer à l'intervalle suivant.

Le numéro et l'intervalle (en secondes) seront réglables dans un fichier de configuration accompagnant l'exécutable.

Schéma


BDD
WEB API
Application
Console
Machine 1
WPF
Application
Console
Machine 2
Application
Console
Machine 3

Architecture C#

L'API sera organisée en 4 couches :

3 couches selon un empilement classique **WEB API** > **Services** > **DAL**

Plus une couche **DTO** de classes servant de contrat de données (paramètres et types de retour pour les méthodes) pour les couches **WEB API** et **Services**.

Cette couche **DTO** sera utilisée également par le WPF et l'application console pour l'envoi et/ou la réception de données à l'API.

Pour rappel, vous pouvez facilement générer le code c# client d'une API avec un outil comme NSwagStudio (https://github.com/RicoSuter/NSwag/wiki/NSwagStudio)

Tests unitaires

Dans votre solution pour l'API, vous ajouterez un projet de tests unitaires xUnit pour l'ensemble des classes et méthodes de la couche **Services**.

Gestion de projet

Le projet doit être géré avec Azure Devops dans lequel vous m'invitez (alan.ferronniere@reseau-cd.net) en vérifiant que j'ai un niveau d'accès « basic » et pas « stakeholder » (dans Organisation settings > Users)

Vous devez veiller à gérer votre projet proprement : pas de push dans master, et des stories liées à tous vos commits et pull request.

Important : je ne veux pas voir un seul gros commit à la fin du projet ! Je veux voir les étapes de développement et votre organisation dans le découpage des tâches à accomplir.