# INPUT GUARD

Fonction: Protéger le système contre des inputs non conforme aux spécifications demandées.

Analyse:
D'après la documentation fournie, le patron Input Guard s'applique aux systèmes qui possèdent les caractéristiques suivantes:

- Le système est composé de composants distincts, qui peuvent chacun être fautif et qui interagissent entre eux en input/output
- Les erreurs qui peuvent être propagées dans un composant système ont la forme d'inputs erronés. Les erreurs internes (causées par des perturbations électromagnétiques) ne sont pas prises en charge par ce patron car pas sous la forme d'un input erroné.

Notre application correspond à cette description, car il y a de nombreux endroits où l'utilisateur peut entrer du texte, pouvant aboutir a de l'injection de code, etc…
La récupération automatique de l'heure (ntp) et les services REST font aussi partie de l'input récupéré.
Par exemple, l'utilisateur peut configurer la connection wifi de l'appareil, et rentrant des valeurs comme `"><script>alert("hello")</script>`, il peut injecter du code malicieux dans la page, qui sera exécuté lors de l'affichage de la page.
Il est donc nécessaire de protéger les input/output, afin d'éviter ce genre de problèmes.
Le patron Input Guard pourrait donc être utile a notre application.

3 implémentations de ce patrons sont possibles:

- Le garde est implémenté comme un nouveau composant du système, et son nombre est proportionnel aux nombres de points d'accès à protéger. Cet manière rajoute du temps d'exécution et augmente l'utilisation mémoire car l'invocation de garde pour chaque point d'accès est coûteux.
- Le garde est implémenté dans le composant à garder. Cet méthode est coûteuse en temps d'exécution car pour rendre le système robuste, on check l'input à chaque appel du composant à garder.
- Le garde est implémenté dans le composant qui envoie l'input. Le nombre d'appels du garde est proportionnel aux nombres de composant qui envoient des données au composant à garder. Le problème est que les garde ne connaissent pas forcément les spécifications du composant à garder afin de traiter l'input correctement. Les erreurs de communication ne sont pas gérées non plus, car le check de l'input se fait avant l'envoie de celui-ci.

Dans le cadre de notre projet, les inputs sont récupérés/utilisés dans les classes: Web et NTP.
Pour faciliter l'utilisation de ce patron avec le Single Access Point, il est préférable d'utiliser la 1ere méthode. En créant une classe ne contenant que des méthodes statiques, on réduit ainsi l'utilisation mémoire, et cette méthode d'implémentation est préférable à la 2eme, car la classe Web est suffisamment lourde et sa responsabilité n'est pas de nettoyer/valider l'input qu'elle reçoit.

Implémentation:

La création d'une classe InputGuard est nécessaire, son rôle sera de traiter l'input avant de l'envoyer au composant à garder. Elle devra utiliser les spécifications du composant qui recevra l'input traité. Ces spécifications seront contenues dans une classe à part. Deux classes utiliseront l'inputguard, mais leurs spécifications sont les mêmes, car elle sont basées sur le protocole HTTP, ce qui permet d'éviter de créer une classe de spécifications supplémentaire.

# SECURE LOGGER

Fonction: Rassembler les log dans un seul endroit et rendre ses logs impossible à altérer.

Analyse:
Ce patron n'est utile que si:

- On a besoin de log des informations sensibles qui ne doivent pas être accessibles à tous.
- On doit pouvoir être sûr que les données loggées sont entières et n'ont pas pu être altérées par un utilisateur malicieux.
- On veut centraliser le système de log pour améliorer la gestion de l'application.
- On veut crypter les données du log pour en assurer l'intégrité et la sécurité.

Dans notre application, les données sont loggées pour des fins de débuggage, et ne contiennent pas d'informations sensible. Le système de log est déjà bien centralisé (passage par la classe Application). Surtout, l'implémentation de ce patron impliquerait la création d'au minimum 4 classes, ce qui serait trop lourd pour notre carte, avec un apport minime sur le fonctionnement.