**SOLID**

1. **Single responsibility principle (Responsabilité unique)**

Chaque class ne doit avoir qu'une seul et unique tâche. Les différentes taches doivent être découplé en plusieurs class.

une classe ne doit changer que pour une seule raison.

1. **Open / Closed principle (ouvert / fermé)**

Une class ne doit pas changé, elle est donc fermé aux modification cependant la class doit pouvoir être extensible, soit ouvert au extensions.

1. **Liskov substitution principle (substitution de liskov) (Barbara Liskov)**

Une class qui hérité de fonctionnalité doit toujours pouvoir y répondre. L'ajout de classes héritées ne devrait pas entraver le fonctionnement d'un système déjà existant.

1. **Interface segregation principle (ségregation des interface)**

Dans la même logique que la responsabilité unique, une interface doit décrire un comportement unique.

1. **Dependency inversion principle (inversio des dépendances)**

les classes de haut niveau ne devraient pas avoir à changer à cause des modifications apportées aux classes de bas niveau.