**SOLID Principls**

**S - Single Responsibility Principle**

Acest principiu urmareste ca obiectele sa aiba o singura responsabilitate, pentru o mai buna si mai usoara intretinere a acestora si de asemenea ca, clasele sa aiba un singur motiv de a se modifica.

**O - Open closed Principle**

Urmareste extinderea claselor fara a le modifica. Putem face asta cu ajutorul abstractizarii, modificand implementarile unor abstractii care inglobeaza o clasa, modificam comportamentul acesteia fara a-i modifica codul.

**L - Liskov Substitution Principle**

Ne sugereaza sa folosimt preconditii si postconditii pentru a nu fi nevoiti sa incalcam Open Closed Principle, in cazul in care o clasa care extinde alta clasa, chiar daca clasa extinsa foloseste alte metode, ea trebuie sa o poata inlocui pe clasa pe care o mosteneste.

**I  -  Interface Segregation Principle**

Acest principiu ne spune ca trebuie sa optam pentru multe interfete mici in schimbul a mai putine interfete stufoase, in cazul in care un client care depinde de anumite utilizari este pus in situatia de a le confunda cu responsabilitati.

**D - Dependency Injection**

Ne spune ca nu ar trebui sa folosim referinte spre clase concrete ci doar spre intefete si/sau clase abstracte. Injectarea se poate folosi pentru a nu fi nevoie sa declaram obiecte concrete in constructor si doar sa folosim abstractii de nivel inalt.