Roşca Alexandru-Mircea

TEMA 1 PSSC

Principiile SOLID

1. **Single-responsibility Principle**: o clasă trebuie să aibă o singură responsibilitate, ea fiind nevoită să îndeplinească o singură funcţie
2. **Open-closed Principle**: obiectele trebuie sa fie deschise pentru extensie, dar închise pentru modificare. O clasă trebuie sa fie uşor extensibilă fără a modifica clasa însine.
3. **Liskov substitution principle**: orice subclasă trebuie sa poată înlocui clasa de bază
4. **Interface segregation principle**: un client nu trebuie să fie obligat să implementeze o interfaţă pe care nu o foloseşte sau să depindă de metode pe care nu le folsesc. Interfeţele „subţiri", cu mai puţine metode, ar trebui alese in locul interfeţelor „grase", care conţin mai multe metode.
5. **Dependency Inversion principle:** Modulele de nivel înalt nu trebuie să depindă de modulele de nivel scăzut, în schimb trebuie să depindă de abstractizare. Ar trebui să avem obiecte de tipul interfeţelor care să ne ajute să comunicăm cu clasele concrete.

Architectura MVC

Model View Controller (MVC) e folosit pentru implementarea interfeţelor cu utilizatorul. E o alegere populară pentru aplicaţiile WEB. Separă logica aplicaţiei în 3 părţi separate, promovând modularitatea, uşurinţa în colaborare şi refolosirea. Face deasemenea aplicaţia să fie mai flexibilă şi mai deschisă spre noi iteraţii.

1. **The Model**
   Modelul defineşte ce date ar trebui să conţină aplicaţia. În cazul în care starea datelor se schimbă modelul va notifica View-ul(astfel interfaţa grafică putându-se schimba în funcţie de nevoie) şi câteodată şi Controller-ul(dacă este necesară o logică diferită pentru a putea controla view-ul actualizat)

2. **The view**

   View-ul defineşte cum ar trebui sa fie afişate datele aplicaţiei şi permite deasemenea modificarea datelor de către utilizator.

3. **Controller**
   Controller-ul se ocupă de cererile utilizatorului. De obicei, utilizatorul interacţionează cu View-ul care trimite request-ul URL potrivit către Controller. Controller-ul alege View-ul potrvit cu datele transmise ca răspuns la request-ul primit.

La începutul WEB-ului, architectura MVC era implementată cel mai mult în partea de server, client-ul trimite request-uri sub forma de forme sau link-uri şi primea view-ul pentru a-l putea afişa înapoi în browser. În ziua de azi, tot mai mult din partea de logică se află în interiorul client-ului.

Exemple de framework-uri ce implementează architectura MVC: AngularJS, Ember.js, Backbone.