**Cuprins**

1. Definirea si justificarea design pattern-urilor implementate
2. Definirea si detalierea metodelor testate prin Unit Testing
3. Definirea si descrierea Test Case-urilor
4. Descrierea sumara a functiilor aplicatiei cu referire la pattern-uri si la metodele testate

Proiect Calitate si Testare Software

Costea Andra Maria

An III, seria A, grupa 1066

Academia de Studii Economice Bucuresti

Facultatea de Cibernetica, Statistica si Informatica Economica

**1. Definirea si justificarea design pattern-urilor implementate**

**1.1 Singleton**

Unul dintre design pattern-urile implementate este Singleton. Acesta este utilizat cu precadere pentru a restricționa numărul de instanțieri ale unei clase la un singur obiect, si astfel reprezintă o metodă de a folosi o singură instanță a unui obiect în aplicație. Acest pattern este utilizat des în situații în care avem obiecte care trebuie accesate din mai multe locuri ale aplicației precum obiectele care reprezintă resurse partajate.

In aplicatia prezentata, Singleton a fost folosit pentru accesul la baza de date. Prin implementarea acestuia se asigura faptul ca se poate instantia un singur obiect de tip DBConnection, care poate fi folosit in mai multe locuri din aplicatie.

**1.2 Observer**

Design Pattern-ul *Observer* se bazeaza pe o relație de dependență 1 la n între obiecte astfel încât când un obiect își schimbă starea, toți dependenții lui sunt notificați și actualizați automat. Pentru a implementa acest design pattern este nevoie de un obiect in rolul de subiect (Subject) si o lista de obiecte care depind de acesta care au rolul de observatori (Observers). Obiectele din aceasta lista sunt parcurse de fiecare data cand se intampla o anumite actiune si se trimit notificari automat.

Acest pattern este unul de tip comportamental dat fiind faptul ca facilitează o organizare mai bună a comunicației dintre clase în funcție de rolurile/comportamentul acestora.

Motivatia folosirii acestui pattern in aceasta aplicatie se datoreaza faptului ca utilizatorii inregistrati isi manifesta interesul pentru proiectele postate pe site. Pentru a le oferi facilitatea de a fi la curent cu noile proiecte incarcate, acestia pot sa primeasca notificari pe email cu informatiile noi. Cu alte cuvinte, in cazul acesta proiectele sunt Subiectul, iar utilizatorii sunt Observerii.

**1.3 Builder**

Design Pattern-ul Builder este unul de tip creational deoarece priveste modul de creare al obiectelor. Alaturi de Abstract Factory, Builder reprezintă obiecte a căror unică responsabilitate este crearea de alte obiecte. Cu alte cuvinte, acest design pattern ajuta la instantierea facila a unor obiecte dintr-o clasa.

In acest proiect, Builder-ul a fost folosit pentru a construi un obiect de tip Mail. Pentru a trimite un email este nevoie de mai multe informatii continute in obiectele clasei Mail. Pentru a construi in-line un obiect din aceasta clasa, am definit o noua clasa MailBuilder care contine o metoda de consctructie efectiva (buildMail) a unui obiect de tip Mail. Aceast tip de design pattern imbunatateste si claritatea codului sursa care devine astfel mai usor de parcurs si depanat.

**1.4 Façade**

Façade este un design pattern structural deoarece se referă la compoziţia claselor sau a obiectelor. Scopul folosirii acestuia este sa ascunda utilizatorului functionalitatile complexe din mai multe clase. Astfel, aceasta noua clasa care se vac rea (Facade) va ingloba apeluri catre mai multe functii continute de diverse clase ale aplicatiei.

In cazul de fata, am folosit Façade pentru a trimite un mail, dar numai catre un utilizator valid. Astfel se apeleaza si metodele de validare ale unui utilizator din clasa User, dar si metodele associate clasei Mail pentru trimiterea unui email. Cu alte cuvinte, am inglobat in functia din Façade numita sendMailToValidUser() toate functionalitatile necesare pentru a face acest lucru cu un singur apel.

**2. Definirea si detalierea metodelor testate prin Unit Testing**

**1. Clasa Project**

Avand in vedere faptul ca un proiect are nevoie de data de inceput si data de final pentru a fi definit, am creat metode care sa permita lucrul cu tipul DateTime. Metodele checkDateFormat si convertDate sunt legate intre ele avand in vedere ca prima verifica daca data furnizata de utilizator este in formatul “ddMMYYYY” acceptat, iar a doua, in cazul in care formatul este bun, o converteste din String in DateTime pentru a putea fi stocata in baza de date si intr-un obiect de tip Project. De asemeni, metoda checkDateStartEnd se refera tot la data si verifica daca data de final furnizata pentru un proiect este dupa data de inceput a acestuia pentru a asigura coerenta in timp.

Clasa mai contine si metode de inserare a obiectelor de tip Project in baza de date InsertIntoDB(), dar si metode specific design patternului Observer. Astfel, am dorit testarea si a faptului ca utilizatorii sunt adaugati in lista pentru a fi notificati prin metoda add().

**2. Clasa User**

Clasa User contine metode in principal care fac referire le informatiile din baza de date. Se doreste filtrarea acestora inaintea inregistrarii in tabele si astfel, se folosesc metode precum cele de verificare a numelui furnizat de utilizator (sa contina numai litere si eventual spatii) si a adresei de email. In acelasi timp, exista si o metoda de inserare a unui utilizator in baza de date la inregistrare. Metoda Update() verifica daca s-au inserat mai mult de 3 proiecte in ultimele trei zile in baza de date si, daca da, trimite mail catre toti utilizatorii abonati la acest eveniment. In aceasta clasa exista si metode de interogare a bazei de date precum cea care intoarce rolul unui utilizator in functie de numele sau (care este unic in baza).

**3. Clasa DBConnection**

Prin clasa aceasta am implementat design patternul Singleton si de aceea am definit metoda getDbInstance care creeaza o noua instanta clasei in cazul in care nu mai exista deja una creata. Metoda GetDBConnection() creaza si intoarce o noua conexiune catre baza de date care va fi folosita in continuare pentru toate tranzactiile facute pe baza.

**4. Clasa Mail**

Aceasta clasa contine metode de verificare a corectitudinii adresei de email furnizate validateAddress(), de trimitere a unui mail in cazul in care adresa este una corecta si de verificare daca doua instante ale acestei clase sunt egale. Metoda insertMailDB inceara inserarea unui obiect de tip Mail in baza de date pentru a tine o evidenta a mesajelor transmise de pe server.

**3. Definirea si descrierea Test Case-urilor**

**1. Clasa Project**

Prin metoda de test testDateFormat() am testat daca data furnizata de utilizator e in formatul care trebuie. Functia initial intoarce un rezultat Boolean, deci am folosit metoda Assert.isTrue pentru a verifica acest rezultat. Prin metoda testDateConversion() am testat daca se converteste string-ul primit in datetime. Am folosit metoda Assert.IsInstanceOf pentru a verifica daca obiectul intors este de tipul DateTime.Prin metoda testDateOrder() am verificat daca data start e mai mica decat data end, pasand doua stringuri ce au fost intai convertite la DateTime si apoi comparate. Prin testUserList() se verifica daca se adauga un user in lista retinuta de Subiect. Am folosit metoda Assert.IsNotEmpty pentru a verifica daca s-a adaugat obiectul in lista care initial era goala. Metoda testInsertIntoDB() testeaza daca se insereaza in baza de date un obiect de tip Project cu success. Datele nu vor fi efectiv inserate in baza ci se va face doar o simulare folosind un obiect de tip TransactionScope.

**2.Clasa User**

Metoda testUpdate() verifica daca se trimit mailuri prin functia specifica patternului Observer catre o anumita adresa de email furnizata. Prin metoda testInsertUser() se testeaza daca se insereaza un obiect de tip user in baza de date folosindu-se obiect de tip TransactionScope. Metoda testGetRole() testeaza daca se intoarce correct rolul userului asociat din baza de date. Rolul intors va fi un String cu numele rolului pe care userul il are retinut in baza. Metoda testVerifyLogin() verifica existent unui utilizator in baza de date dupa username-ul si parola furnizate de acesta. Aceasta functie este folosita la inregistrarea in aplicatie. Prin testVerifyMailAddress() se testeaza daca adresa mail a userului este una valida din punct de vedere al sintaxei, insa nu si daca aceasta este asociata cuiva efectiv. Cele doua metode testVerifyUsername() si testVerifyUsernameSpaces() verifica daca numele utilizatorului este compus numai din litere in primul caz si, in al doilea, daca acesta contine si spatii oentru a delimita numele de prenume sau prenumele intre ele.

**3. Clasa DBConnection**

Prin ConnectToDbInstance() am testat daca se face conexiunea la baza de date prin obiectul de tip DBConnection, iar prin metoda CreateDbInstance() se testeaza daca se creaza o noua instant a clasei.

**4.Clasa Mail**

Prin metoda testMailSend() am verificat daca se trimite mailul in conditii optime si cu parametrii corespunzatori. Folosind metoda testMailSendAddress() am testat daca se trimite mailul in cazul in care I se furnizeaza o adresa incorecta. Prin metoda testPassEncryption() am testat faptul ca parola furnizata este diferita de cea rezultata in urma criptarii. Prin testMailBuilderInstance() am verificat daca se creaza un obiect nou de tip mail folosind crearea obiectului mail prin patternul builder. Prin testMailBuilderValue() am testat daca se creaza un nou obiect care este de tipul Mail in urma folosirii Builderului. Prin metoda testInsertMailDb()testez daca se insereaza obiectul de tip Mail in baza de date cu success.

**4. Descrierea sumara a functiilor aplicatiei cu referire la pattern-uri si la metodele testate**

Prin folosirea patternurilor discutate pe parcursul acestei lucrari, aplicatia a obtinut noi functionalitati. Una dintre acestea se refera la faptul ca atunci cand administratorul introduce cel putin 3 proiecte noi in decursul ultimelor trei zile, toti utilizatorii vor primi un mail de instiintare cu informatii despre aceste noutati. Adaugarea se poate face prin formularul AddProject si verificarea in baza si tirmiterea automata a mailurilor se face prin apasarea butonului de adaugare. Acesta permite si adaugarea in baza de date a proiectului in cazul in care datele introduce sunt validate si considerate corecte prin utilizarea functiilor de verificare din clasele associate. In acelasi timp, aplicatia permite inregistrarea de noi utilizatori care vor fi adaugati dupa validare in baza de date. In cazul utilizatorilor care au deja cont creat, acestia se pot loga in aplicatie si vor fi redirectionati catre pagina lor personala printr-o verificare in baza de date a faptului ca acestia sunt inregistrati.

Bibliografie:

<https://msdn.microsoft.com/en-us/library/hh694602.aspx>

<http://stackoverflow.com/questions/10168240/encrypting-decrypting-a-string-in-c-sharp>

<http://getbootstrap.com/>

https://csharpdesignpatterns.codeplex.com/