

# **Jurnal Securizat**

Bolboceanu Andrei Răzvan

Universitatea de Vest din Timișoara, Bv. Vasile Pârvan nr 4, Timișoara 300223,  
Romania

## Table of Contents

Jurnal Securizat .....	1
<i>Bolboceanu Andrei Răzvan</i>	
1 Introducere .....	3
2 Tehnologiile folosite .....	3
3 Securizarea aplicatiei .....	4
3.1 Login .....	7
3.2 Crearea de noi jurnale .....	9
3.3 Modificare Jurnal .....	12
4 Interfata grafica .....	12

## 1 Introducere

Securitatea a ajuns in zilele noastre o chestiune foarte importanta, dar care din pacate nu este luata in seama de cei mai multi. Schimbul de informatii intre persoana a ajuns sa fie atat de des incat nu ne mai putem imagina o lume fara acesta. Dar cati dintre noi suntem constineti de securitatea convorbirilor, cati dintre noi suntem constienti de securitatea datelor mai sensibile cum ar fi datele bancare. Asa ca pentru proiectul de fata ne-am propus sa implementam o aplicatie de jurnal securizata, folosind ca si limbaj de progrmare java si librarii criptografice aferente. Pe scurt aplicatia are in spate o baza de date, care va fi accesata de un utilizator printr-un proces de inregistrare, ca mai apoi acesta sa isi poata crea sau modifica jurnale, toate acestea cu ajutorul unui interfete grafice.

## 2 Tehnologiile folosite

Pentru crearea aplicatiei de jurnal securizata s-a folosit limbajul java in Eclipse, interfata grafica a fost creata folosind java.Swing, si pentru partea de securizare a aplicatiei s-a folosit in principal libraria java.security. Pentru stocare utilizatorilor s-a folosit un server localhost folosind aplicatia xampp, si o baza de date MySQL. Baza de date este compusa din doua tabele, primul care retine utiliza-

Fig. 1: Schema bazei de date

test2 jurnal	test2 users
id_jurnal : int(11)	id : int(255)
# id_user : int(11)	username : varchar(255)
nume_jurnal : varchar(255)	password : varchar(255)
continut : varchar(255)	salt : varchar(255)
salt : varchar(255)	date : timestamp
date : timestamp	

torii si cel de-al doilea tabel contine jurnalele. Dupa cum se poate observa in imaginea de mai sus tabelul de users are urmatoarele campuri:

- id-ul care este primary-key si se autoincrementeaza in momentul in care un utilizator nou se inregistreaza
- username de tip varchar care retine numele utilizatorului
- password de tip varchar care va retine parola hash-uita
- salt de tip varchar care va retine sarea folosita la hashuirea parolei
- date de tip timestamp care indica data si ora cand un utilizator nou s-a inregistrat in aplicatie

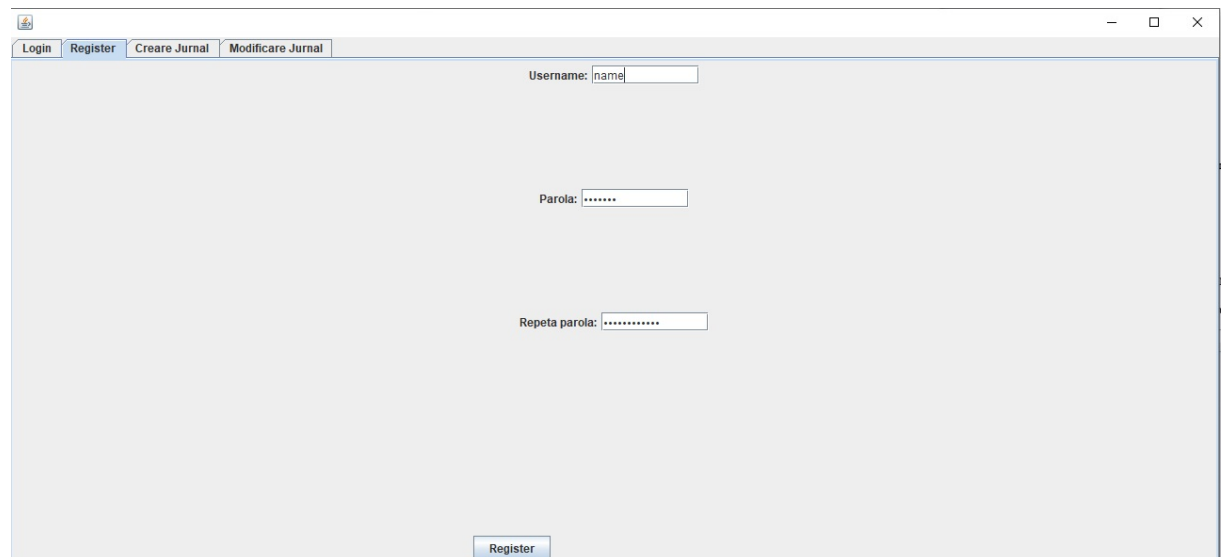
Cel de-al doilea tabel este compus din:

- id-ul jurnalului care este primary-key autoincrementat
- id-ul userului de tip int care retine id-ul userului proprietar jurnalului
- titlul jurnalului de tip varchar care retine numele jurnalului
- continutul de tip varchar care va retine continutul jurnalului, acest camp este criptat
- salt de tip varchar care retine sarea cu care continutul jurnalului a fost criptat
- date de tip timestamp care inregistreaza data si ora in care jurnalul a fost creat

### 3 Securizarea aplicatiei

Pentru a avea acces la aplicatie si pentru ca un utilizator sa poate crea jurnale acesta trebuie sa se inregistreze.

Fig. 2: Formular inregistrare(interfata)



The image shows a web application window with a title bar and four tabs: 'Login', 'Register', 'Creare Jurnal', and 'Modificare Jurnal'. The 'Register' tab is active. The form contains three input fields: 'Username: [name]', 'Parola: [password]', and 'Repeta parola: [password]'. All password fields are masked with dots. A 'Register' button is located at the bottom center of the form area.

Pentru a se intrregistra trebuie sa completeze formularul de inregistrare din figura2. In acest panel intalim 3 spatii de text si un buton. Primul spatiu este dedicat numelui, celelalte doua saptii sunt dedicate parolei respectiv repetarea acesteia. Pentru o mai buna confidentialitate a datelor s-a ales folosirea JPasswordField. Acest camp preia numele utilizatorului ca un byte array care mai apoi va fi convertit in string si inserat in baza de date. Username-ul ales va trebui sa fie unic. Parola trebuie sa fie la fel in ambele campuri de parola. In caz contrar se va afisa in consola un mesaj de tip Logger cu notificarea aferenta erorii. Dupa ce datele introduse sunt corecte utilizatorul va apasa pe butonul de Register si acesta va fi inserat in baza de date.

```

1 private final static String ALGORITHM="SHA512";
2 static byte[] salt=null;
3
4 public static byte[] GnerateSalt() throws
   NoSuchAlgorithmException{
5
6     SecureRandom sc=SecureRandom.getInstance("SHA1PRNG");
7     byte[] salt=new byte[16];
8     sc.nextBytes(salt);
9     return salt;
10 }
11
12 public static byte[] Hashing(String password,byte[] salt)
   throws NoSuchAlgorithmException {
13     MessageDigest dg=MessageDigest.getInstance(ALGORITHM);
14     dg.reset();
15     dg.update(salt);
16     byte[] hash=dg.digest(password.getBytes());
17     return hash;
18 }
19
20 public static String bytesToHex(byte[] bytes) {
21     String s = Base64.getEncoder().encodeToString(bytes);
22     return s;
23 }
24
25 public static byte[] hexStringToByteArray(String hex) {
26     byte[] decode = Base64.getDecoder().decode(hex);
27     return decode;
28 }
29

```

Listing 1.1: Clasa Encrypting

```

1 public static void register(String username,String password)
   throws NoSuchAlgorithmException {
2     PreparedStatement ps;
3     byte[] byteSalt=null;
4     byteSalt=Encrypting.GnerateSalt();
5     byte[] pass=Encrypting.Hashing(password,byteSalt);
6     String passToHex=Encrypting.bytesToHex(pass);
7     String saltToHex=Encrypting.bytesToHex(byteSalt);
8
9     try {
10         String sql="INSERT INTO users(username,password,
   salt) VALUE (?,?,?)";
11         ps=DbConnect.getInstance().getConnection().
   prepareStatement(sql);
12         ps.setString(1, username);
13         ps.setString(2, passToHex);
14         ps.setString(3, saltToHex);
15         if(duplicate(username)) {
16             ps.executeUpdate();
17             Main.logger.info("Inserare user reusita!");}
18         else
19             Main.logger.info("Username-ul este deja luat!");
20
21     } catch (SQLException e) {
22         // TODO Auto-generated catch block
23         e.printStackTrace();
24     }
25 }
26
27





```

Listing 1.2: Functia de inregistrare din clasa User

Cand utilizatorul va apasa pe butonul de Register se va apela functia de inregistrare din Listing 1.2. Pentru hashuire parolei s-a generat o sare folosind functia GenerateSalt(). Aceasta functie genereaza aleator un byte array de 16 biti folosind SecureRandom prin "SHA1PRNG" ca generator de numere aleator. Functia hashing() va lua ca parametru parola utilizatorului si o sare generata aleator. Acesta foloseste MessageDigest care este o functie hash criptografica care contine un sir de cifre create printr-o formula hash unidimensionala. In aceasta functie se creaza un obiect de tipul MessageDigest si prin metoda getInstance() ofera un obiect care aplica algoritmul criptografic dorit, in cazul nostru SHA512. Functia mai apoi updateaza obiectul cu un byte array in cazul nostru sarea si prin digest() genereaza un byte array folosind sarea si parola introdusa de utilizator. Acest byte array este de fapt parola hash-uita. Ultimele doua functii byteToHex() si hexStringToByteArray() au rolul de a transforma hash-ul rezultat din functia de Hashing si sarea in String, respectiv de a transforma din string in byte array. Trecand la functia de Register (listing 1.2) aceasta are urmatorul mod de functionare:

- preia username-ul introdus si parola
- se genereaza o sare folosind functia GenerateSalt()
- aceasta sare generata impreuna cu parola vor fi date ca parametru functiei de hashing() din care va rezulta parola hash-uita sub forma de byte array.
- byte array-ul de hash si sarea vor fi transformate in string
- se verifica daca username-ul este deja existent si in caz contrar se va adauga un nou utilizator in baza de date.

Fig. 3: Utilizatorii in baza de date

				id	username	password	salt	date
<input type="checkbox"/>	 Edit	 Copy	 Delete	20	nume7	zMTMnLaHPf/iU9kS2Bg3qA==	09vUwfOlPrxOZ2dqQYNv9w==	2021-05-16 23:19:57
<input type="checkbox"/>	 Edit	 Copy	 Delete	21	nume1	bG6RfetQgr4eF+eGskHpCA==	QotxgjREtFvh09NKE/3WA==	2021-05-16 23:36:20

Pentru a preveni eventualele injectii SQL se foloseste PreparedStatement. Majoritatea acestor injectii SQL au loc deoarece interogările sunt luate ca text.

### 3.1 Login

Dupa ce utilizatorul s-a inregistrat acesta trebuie sa acceseze panel-ul de login din figura 4.

Acest formular este compus din doua capuri de text si anume username si parola. Parola la fel ca si in formularul de inregistrare este luata ca un byte array si convertita in string.

```

1  public static boolean getValid(String username,String
    password) throws IOException, NoSuchAlgorithmException,
    SQLException {
2      PreparedStatement ps;
3      try {
4          String sql="SELECT salt,password FROM users WHERE
    username= ?";
5          ps=DbConnect.getInstance().getConnection().
    prepareStatement(sql);
6          ps.setString(1,username);
7          ResultSet result=ps.executeQuery();
8          if(result.next())
9              {
10                 byte[] byteSalt=Encrypting.hexStringToByteArray(
    result.getString(1));
11                 byte[] loginPass=Encrypting.Hashing(password,
    byteSalt);
12                 byte[] storedPass=Encrypting.hexStringToByteArray
    (result.getString(2));

```

```

13         boolean res=Arrays.equals(loginPass, storedPass);
14         if(res) {
15             Main.logger.info("logare reusita");
16             return true;
17         }
18     }
19     }catch (NoSuchAlgorithmException e) {
20         e.printStackTrace();
21     }
22
23     System.out.println("Datele sunt incorecte");
24     return false;
25 }

```

Listing 1.3: Functia de login din clasa User

Fig. 4: Formular inregistrare(interfata)

Functia `getValid()` se ocupa cu autentificare unui utilizator. Aceasta functie preia datele din panelul de login (figura 3) si pe baza acestora verifica datele utilizatorului. Pe baza datelor introduse extrage din baza de date sarea aferenta utilizatorului, mai apoi aceasta fiind de tip string o va transforma in byte array. Parola introdusa de catre utilizator impreuna cu sarea este data ca parametru functiei de `hashing()` care va genera pe baza acestora un hash. Acest hash este comparat cu cel introdus in baza de date. Daca aceste doua coincid userul si-a



introdus datele corect si este lasat sa isi continue treaba pe platforma, in caz contrar se va afisa in consola un mesaj de tip logger.

### 3.2 Crearea de noi jurnale

Dupa ce utilizatorul s-a logat in aplicatie acesta va putea sa creeze noi jurnale accesand panelul Create Jurnal. Un jurnal e compus din titlu si continut.

Fig. 5: Panel creare jurnal

Cand utilizatorul se logheaza, se va seta un id pentru ca jurnalul introdus sa fie atribuit persoanei logate. Daca persoana nu se logheaza si acest id nu este setat utilizatorul nu va crea un jurnal nou.

```

1      public static String encrypt(String strToEncrypt, byte[]
2      salt) {
3          try {
4              IvParameterSpec iv = new IvParameterSpec(salt);
5              String SaltToString=Encrypting.bytesToHex(salt);
6              SecretKeyFactory factory = SecretKeyFactory.
7              getInstance("PBKDF2WithHmacSHA1");
8              KeySpec spec = new PBEKeySpec(SaltToString.
9              toCharArray(), salt, 65536, 256);
10             SecretKeySpec secretKey = new SecretKeySpec(factory.
11             generateSecret(spec).getEncoded(), ALGORITHM);

```

```

9      Cipher cipher = Cipher.getInstance("AES/CBC/
PKCS5Padding");
10      cipher.init(Cipher.ENCRYPT_MODE, secretKey, iv);
11      return Base64.getEncoder().encodeToString(cipher.
doFinal(strToEncrypt.getBytes(StandardCharsets.UTF_8)));
12  } catch (Exception e) {
13      System.out.println("eroare la criptare " + e.toString
());
14  }
15      return null;
16  }
17
18  public static String decrypt(String strToDecrypt, byte[] salt
) {
19      try {
20          IvParameterSpec iv = new IvParameterSpec(salt);
21          String SaltToString=Encripting.bytesToHex(salt);
22          SecretKeyFactory factory = SecretKeyFactory.
getInstance("PBKDF2WithHmacSHA1");
23          KeySpec spec = new PBEKeySpec(SaltToString.
toCharArray(), salt, 65536, 256);
24          SecretKeySpec secretKey = new SecretKeySpec(factory.
generateSecret(spec).getEncoded(), ALGORITHM);
25
26          Cipher cipher = Cipher.getInstance("AES/CBC/
PKCS5PADDING");
27          cipher.init(Cipher.DECRYPT_MODE, secretKey, iv);
28          return new String(cipher.doFinal(Base64.getDecoder().
decode(strToDecrypt)));
29      } catch (Exception e) {
30          System.out.println("eroare la decryptare " + e.
toString());
31      }
32      return null;
33  }

```

Listing 1.4: funcțiile de criptare a conținutului unui jurnal

Cele două funcții au rolul de a encipita și a decipita până la nivel de text deoarece un utilizator în cazul în care dorește să modifice conținutul unui jurnal aplicația trebuie să îi permită să vadă textul decipitat. Conținutul unui jurnal în baza de date este criptat cu o sare generată aleator. Criptarea conținutului se face în funcție `encrypt()` din clasa cheie (listing 1.4). Această funcție ia ca parametru `String`ul ce conține conținutul jurnalului și sarea în `byte` array. Pentru început se creează `iv`-ul apelând constructorul `IvParameterSpec(byte[] iv)`, această clasă specifică un vector de initializare (IV) care va avea rolul în construcția "block cipher mode of operation". Cele mai multe moduri cer o secvență unică binară, în cazul nostru `iv`-ul folosită pentru oricare operație de criptare/decipitare. Acest vector este folosit pentru a se asigura că sunt produse texte criptate diferite chiar dacă se folosește aceleași cheie. După ce `iv`-ul a fost creat vom avea nevoie

de o cheie secreta pe care o vom genera apeland constructorul `SecretKeyFactory` care va returna un obiect de tipul `SecretKeyFactory` care va converti cheile algoritmului specificat `"PBKDF2WithHmacSHA1"`. `"PBKDF2"` provine de la `"password-based-key-derivative-function"` si este folosit pentru implementarea unei functii pseudo aleatorii cum ar fi functia criptografica de hash `"HMAC"`. `"HMAC"` provine de la `keyed-hash message authentication code` si se foloseste pentru calcularea unui cod de autentificare a mesajelor care implica o functie hash criptografica in combinatie cu o cheie secreta. Orice functie de hash criptografica poate fi utilizata in acest calcul, noi folosind `"HMAC-SHA1"`. Mai apoi se va crea un obiect de tipul `KeySpec` apeland constructorul `PBEKeySpec` care va lua ca parametru sarea ca si char array, sarea ca si byte array precum si numarul de iteratii impreuna cu lungimea cheii. Numarul de iteratii reprezinta de cate ori parola este hashuita in timpul derivarii cheii simetrice. Cu cat e mai mare numarul cu atat este mai dificila eventuala ghicire a unei parole. Acest nr de iteratii este folosit impreuna cu sarea. Lungimea cheii este lungimea in biti a cheii simetrice derivate in functie de algoritmul folosit, in cazul nostru se foloseste sha1 deci lungimea cheii va fi de 256biti. Ajungand la finalul partii de generare a cheii secrete se foloseste constructorul `SecretKeySpec` care ia ca si parametru materia cheia encodata in biti si algoritmul ales pt crearea cheii secrete in cazul nostru `"AES"`. Dupa ce s-a generat cheia secreta se trece la partea de criptare in sine. Clasa cipher este o clasa care se ocupa cu acest lucru. Clasa Cipher se instantiaza apeland metoda `getInstance()` si parasandui numele cifrului ca si parametru, in cazul nostru `"AES/CBC/PKCS5Padding"`. Numele cifrului este compus dupa cum se poate observa din trei parti. Prima parte este numele algoritmului in cazul nostru `"AES"`, cea de-a doua parte este reprezentata de modul in care algoritmul trebuie folosit `"CBC"` si ultima parte este schema de padding aleasa in cazul nostru `"PKCS5Padding"`. Mai apoi se initializeaza instanta Cipher tocmai creata apeland metoda `init()` careia trebuie sa ii atribuim trei parametri si anume modul de criptare `ENCRYPT MODE` pentru criptare sau `DECRYPT MODE` pentru decriptare, cheia secreta generata anterior si iv-ul. Pentru ca datele sa se encrpteze vom apela metoda `doFinal()`, o sa encodam textul in baza 64 pentru a ne asigura care nu va suferii modificari la transfer. Functia de encrypt va returna textul criptat sub forma de string care mai apoi va fi introdus in baza de date alaturi de sarea.

Fig. 6: Jurnal in baza de date

	id_jurnal	id_user	nume_jurnal	continut	salt	date
<div> <div>← T →</div> <div>▼</div> </div> <div> <div>□</div> <div>Edit</div> <div>Copy</div> <div>Delete</div> </div>	38	21	jurnal	L'P□□□j mǎ'z□ôo@□	XD XuJ18WfyBKsD8SU4YF7w==	2021-05-22 16:28:56

### 3.3 Modificare Jurnal

Utilizatorul are posibilitatea de a modifica continutul unui jurnal accesand tabul de modificare jurnal prezentat mai jos Acest panel este compus dintr-un meniu de tip dropdown care contine toate jurnalele utilizatorului logat, un text area care contine continutul jurnalului ales decriptat si butonul de modificare jurnal.

```

1
2 public static void modificareJurnal(int userId,int jurnalId
3 ,String continutNou) throws Exception {
4     PreparedStatement ps;
5     byte[] byteSalt=Encripting.GnerateSalt();
6     String prepareContinut=Cheie.encrypt(continutNou,
7     byteSalt);
8     try {
9         String sql="UPDATE 'jurnal' SET 'continut'=?,salt=?
10 WHERE id_jurnal=? AND id_user=?";
11         ps=DbConnect.getInstance().getConnection().
12         preparedStatement(sql);
13         ps.setString(1,prepareContinut);
14         ps.setString(2,Encripting.bytesToHex(byteSalt));
15         ps.setLong(3, jurnalId);
16         ps.setLong(4, userId);
17         ps.execute();
18         Main.logger.info("junral modificat cu succes");
19     }catch(Exception e){
20         e.printStackTrace();
21     }
22 }

```

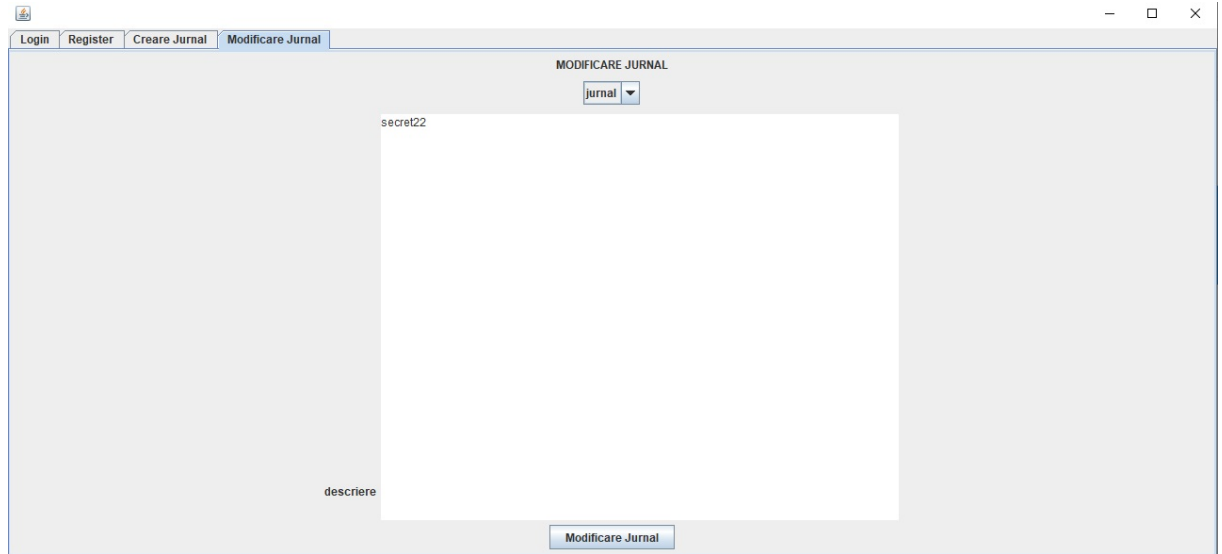
Listing 1.5: functiile de modificare a continutului unui jurnal

Functia de modificareJurnal modifica jurnalul in functie de id-ul utilizatorului si de id-ul jurnalului. In aceasta functie se genereaza o sare folosind functia de GenerateSalt() din clasa Encripting, mai apoi aceasta sare si noul continut sunt criptate folosind functia encrypt din clasa Cheie. Functia encrypt va rezulta un byte array care mai apoi va fi convertit in String, si continutul modificat se va schimba in baza de date. Pentru o mai buna securizare s-a ales ca la fiecare modificare sa se genereze o noua sare si continutul nou sa se cripteze cu aceasta.

## 4 Interfata grafica

Interfata grafica a fost facuta folosind java.swing. Clasa Repository este una din clasele de baza ale aplicatiei, intrucat toate operatiile asupra bazei de date se realizeaza cu ajutorul acestei clase. Aceasta clasa contine lista cu toate obiectele aflate in baza de date. Mai exact lista cu jurnalele. De asemenea o lista foarte importanta este lista de observatori. Repository fiind actorul principal al acestei aplicati, exista o serie de observatori care se modifica in functie de datele noi

Fig. 7: tabul de modificare jurnal



sau modificare din repository.

De exemplu atunci cand se introduce un nou jurnal in aplicatie, datele acestuia sunt preluate din interfata grafica si apoi sunt trimise catre repository. Acesta mai intai va introduce jurnalul in lista de obiecte de tip jurnal, apoi jurnalul va fi salvat in baza de date iar apoi toti observatorii(combobox-uri) vor fi actualizate cu privire la datele noi introduse. Pe langa aceasta clasa fiecare buton si drop-box are clasa lui separata spre exemplu clasa ComboJurnale, aceasta clasa se ocupa doar de meniul drop-down din panelul de modificare jurnal. Aceasta clasa extinde JComboBox adica va avea rolul unui combo-box, aceasta clasa implementeaza Command si Observer deoarece daca utilizatorul este logat si in timpul acesta isi introduce noi jurnale, se va updatea si combo-boxul cu jurnalele noi introduse in timp real.