Proiect MDS

Catalog Online

### 

### 

Membri echipa

Frigioiu Lucian

Melinte Ana-Maria

Mihalachi Lavinia

Sandru Adrian

# Cuprins

[**Cuprins**](#_fpqh94byh2gf) **2**

[**Descriere**](#_9rtbxhkayb28) **3**

[**User stories**](#_pwg5f81mnxqv) **4**

[User story 1](#_99xmnalhj9x3) 4

[User story 2](#_1f4ef5bo3bi8) 4

[User story 3](#_mgfrgk1brahf) 5

[User story 4](#_mgfrgk1brahf) 5

[**Diagrame UML**](#_y04ri682423i)6

[Diagrame Clase](#_udgkj5jm99s) 6

[Diagrame Stari](#_x14cuyl5trbp) 7

[Diagrame Secvente](#_chp9kfqntdde) 8

[Diagrame Cazuri de utilizare](#_p0v152i03z5y) 8

[**Source Control**](#_n2m94oc4pduq)9

[**Teste automate**](#_lrdldf6ynz0n)10

[**Bug report**](#_tlynqrxq2gaa)11

[Bug 1](#_jbyklgj9kyat) 11

[Bug 2](#_p5880zuebwhd) 11

[Bug 3](#_p5880zuebwhd) 12

[Bug](#_p5880zuebwhd) 4 12

[**Refactoring**](#_87zlljj2s2df)13

[**Design patterns**](#_5dprf8x9llf6)15

[Singleton](#_5esmhc1trrxf) 15

# 

# Descriere

Aplicatie desktop destinata liceelor, scopul principal al acesteia fiind facilitarea comunicarii intre parinti – elevi – profesori.

Principalele functionalitati for fi:

-login personalizat (cu username si parola) pentru fiecare utilizator, avand patru tipuri de utilizatori: director, profesor, parinte si elev. Pentru fiecare utilizator dupa login se face redirectioneaza catre o pagina specifica.

-fiecare user isi poate schimba parola

-profesorii:

* pot adauga si motiva absentele elevilor
* pot adauga note

-elevii :

* isi pot dea notele si absentele
* isi pot vedea un overview asupra performantei
* isi pot vedea media finala

-directorii:

* pot actualize lista de elevi

-parintii:

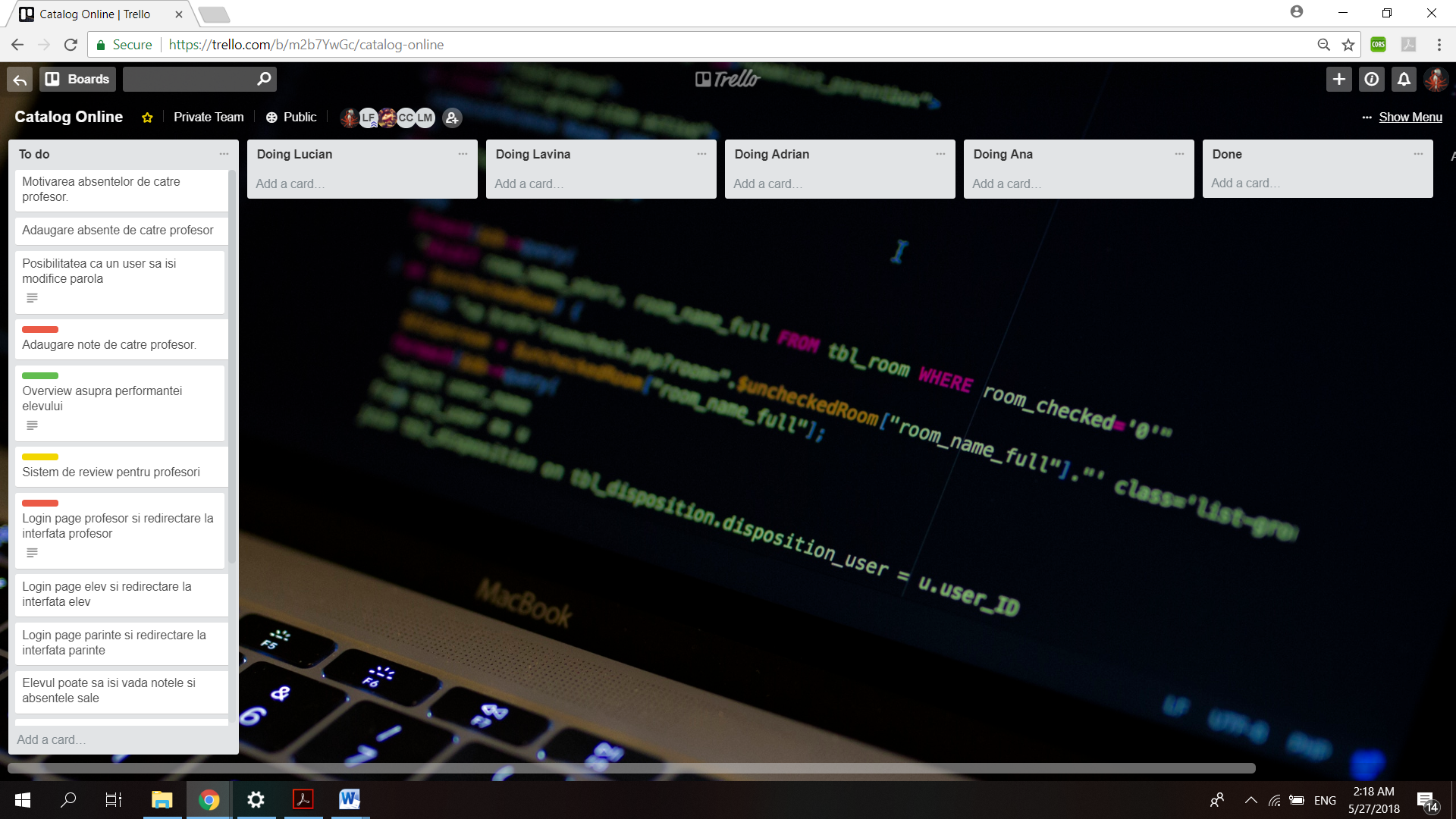
* au acces la notele si absentele elevului

# 

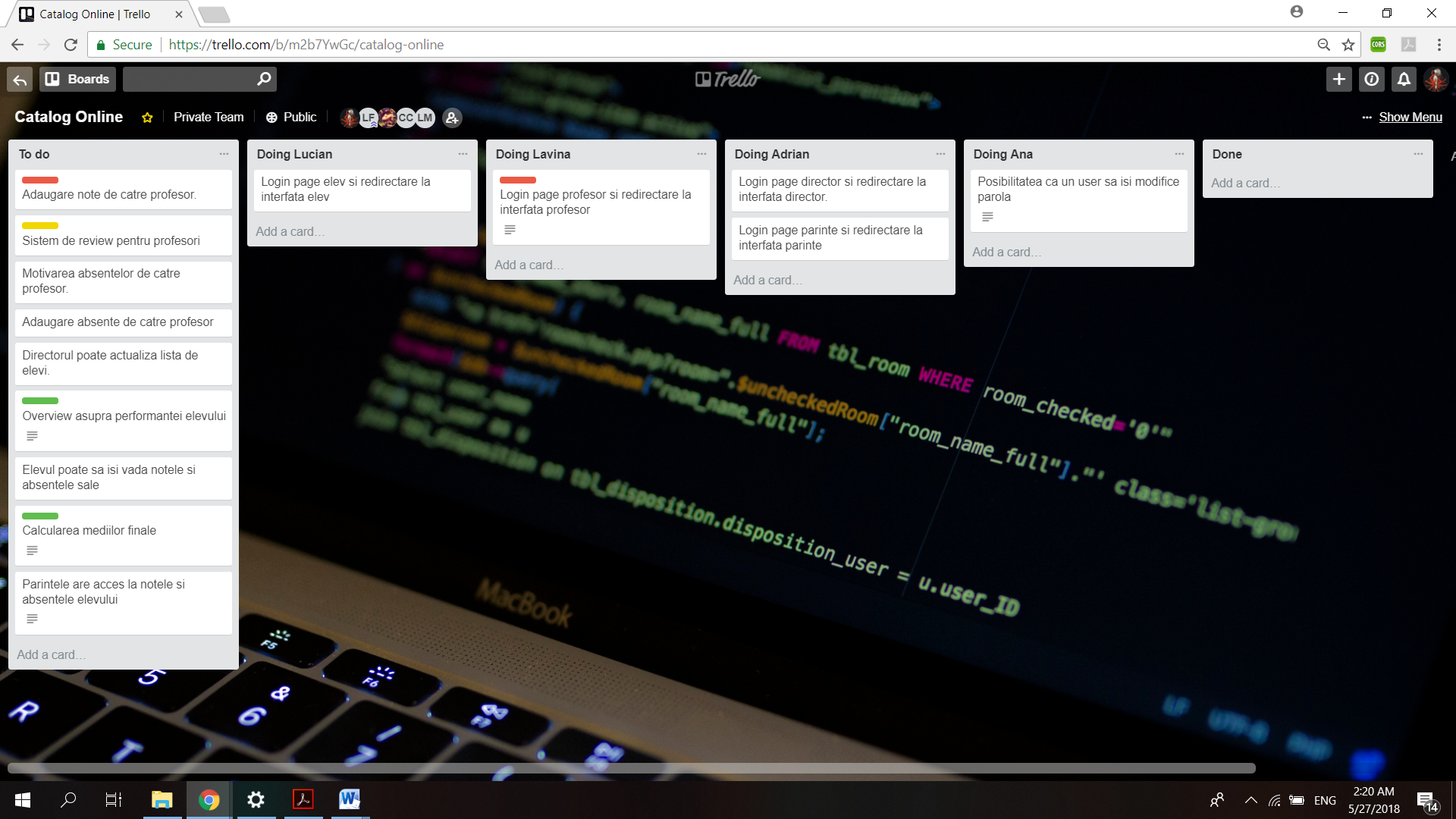
# User stories

https://trello.com/b/m2b7YwGc/catalog-online

### User story 1



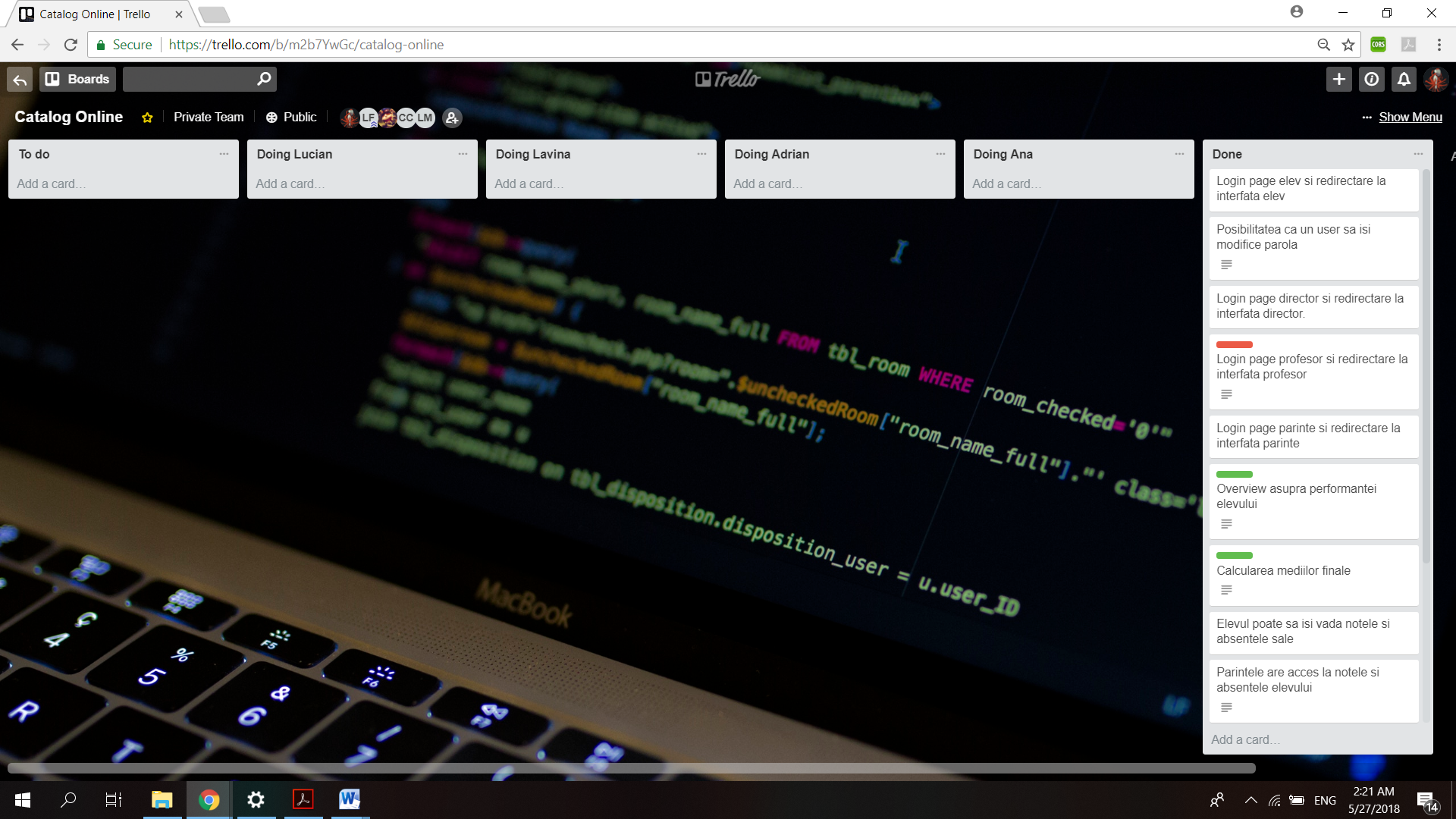
### User story 2



### User story 3



### User story 4



# Diagrame UML

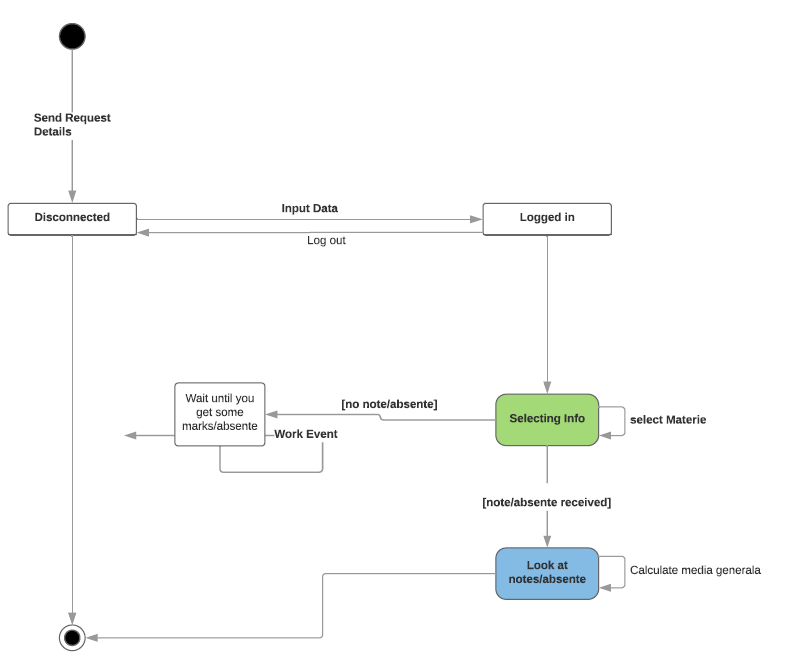
## Diagrame Clase

## 

## 

## Diagrame Stari

## 



## Diagrame Secvente

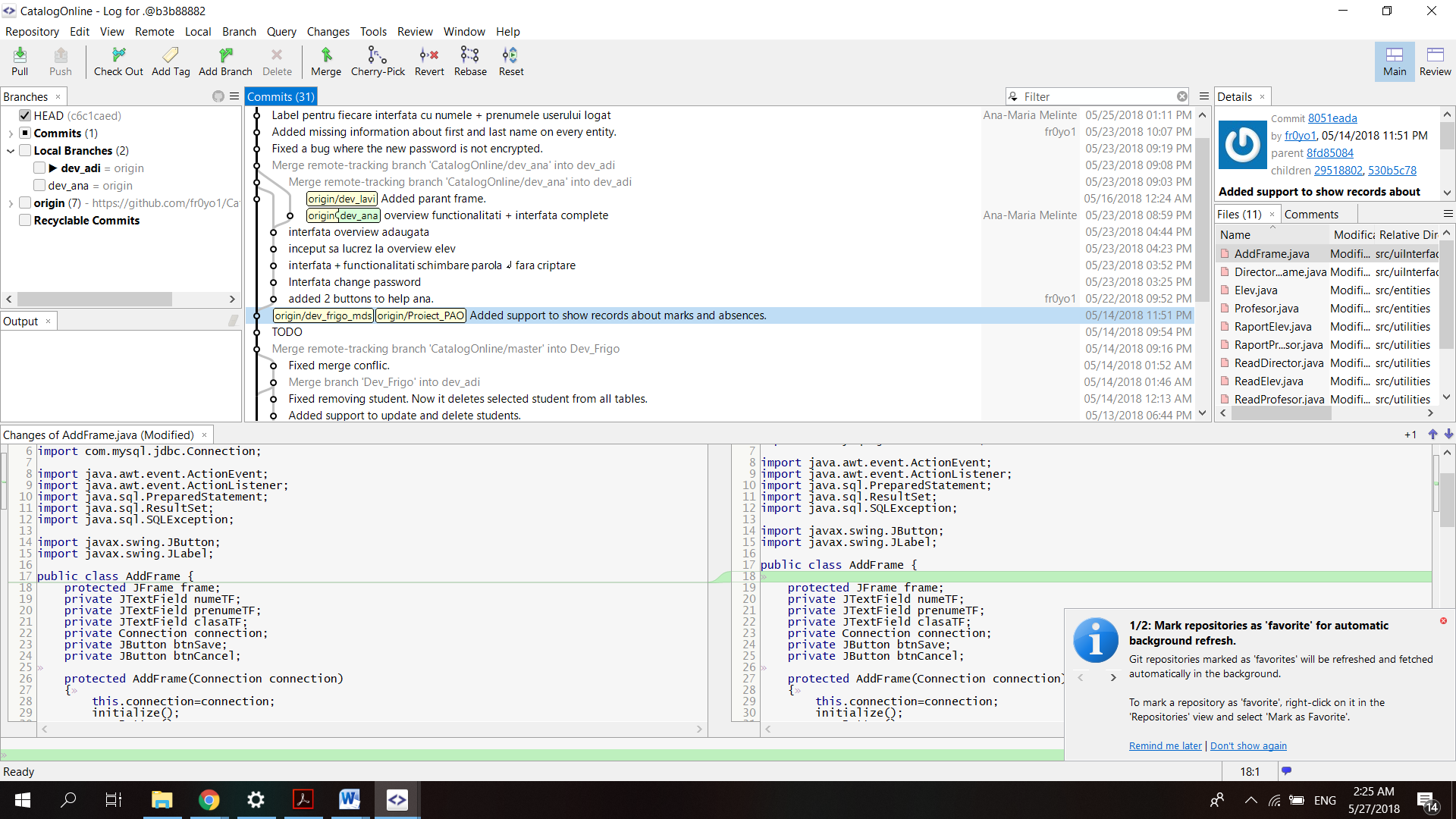
## 

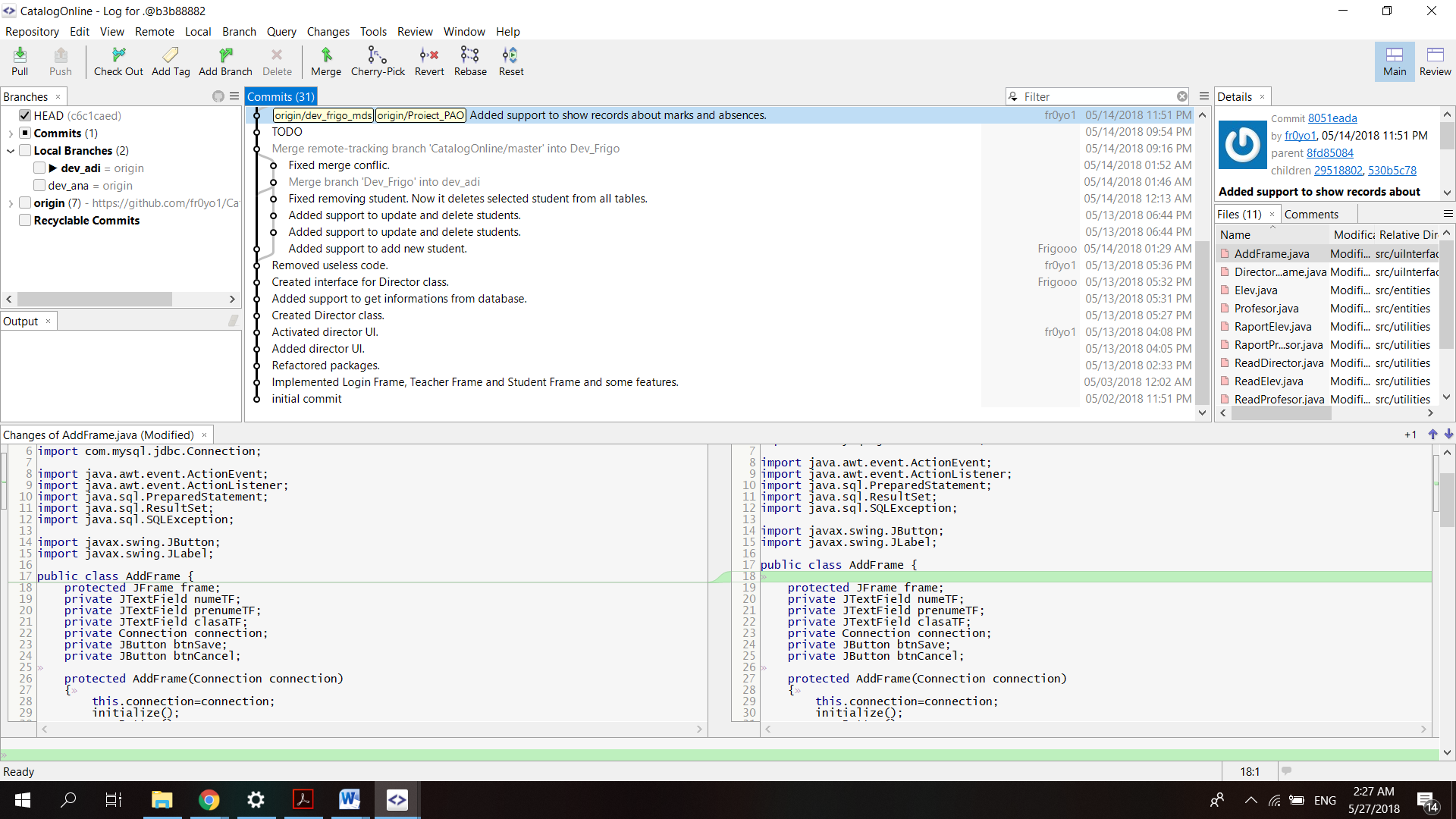
## Diagrame Cazuri de utilizare

# 

# Source Control

<https://github.com/fr0yo1/CatalogOnline>





# 

# Teste automate

Test that Catalog Online works correctly.

Testing Round 1

Test that the login works fine for any type of users

1.Login with sandru.adrian / password

2.Login wih luminita.cobzaru/ password

3.Login with mircea.popescu/password

4.Login with vasile.ciuchina/password

Notice that you are logged in everytime with a different interface.

Testing Round 2

Test that Elev can see his absences and his notes

1.Login with sandru.adrian/password

2.Select a subject from dropdown list .

3.Click GetNote

4.Click getAbsente

Notice that they are listed in the textbox below.

Testing Round 3

Test that Elev can see his level of competency in his class.

1.Login with sandru.adrian/password

2.Select a subject from the dropdown list

3.Click Overview

Notice that you should see the students' level

Testing Round 4

Test that a Teacher can choose a student and see his marks or absences

1.Login with luminita.cobzaru/password

2.Select a student after selecting the class and the subject

3.See that the marks and abences are shown in the list below

Testing Round 5

Test that a teacher can add a new mark or an absence to a specified student

1.Login with luminta.cobzaru/password

2.Select the student after selectin the class and the subject,

3.Select a date from the DatePickerCalendar

4.Add a mark.

5.Add an Absence.

6.Check if the Absence and the mark was added with the today date.

Testing Round 6

Test that the director can see the information about a student

1.Login with vasile.ciuchina/password

2.Select a student

Notice that the textboxes should show the info about the selected student.

Testing Round 7

Test that the director can add/edit/delete a student

1.Login with vasile.ciuchina/password

2.Add a student

3.Check if the student was added in the list.

4.Edit a student

5.Check if the information was changed accordingly.

6.Delete the student.

7.Check if the student was deleted from the list.

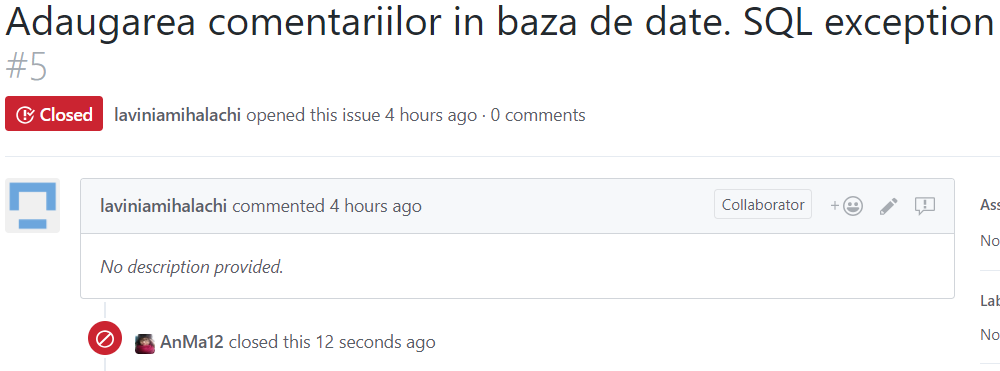
# Bug report

### Bug 1

# Adaugarea comentariilor in baza de date. SQL exception

<https://github.com/fr0yo1/CatalogOnline/issues/5>

tmt = (PreparedStatement) connection.prepareStatement("INSERT INTO comentarii Values (?,?,?);");  
stmt.setString(1, id\_elev); stmt.setString(2, id\_profesor);  
stmt.setString(3, comment); stmt.executeUpdate();



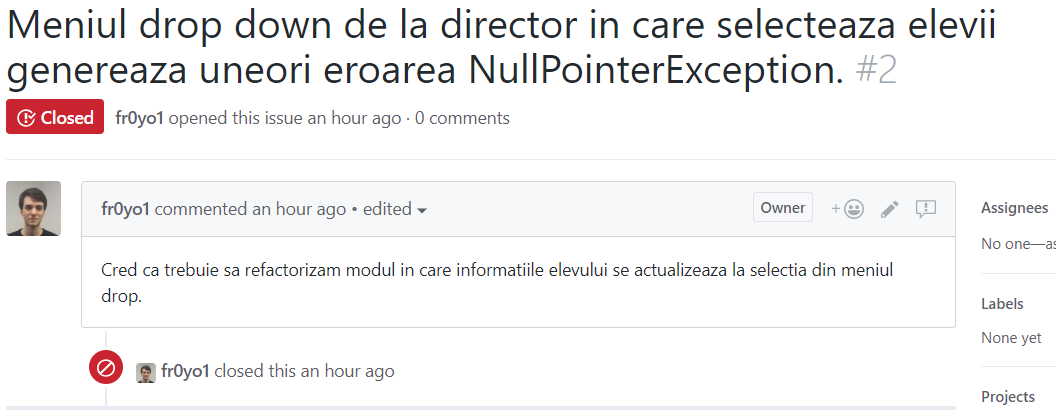
### Bug 2

# Meniul drop down de la director in care selecteaza elevii genereaza uneori eroarea NullPointerException.

<https://github.com/fr0yo1/CatalogOnline/issues/2>

Problema a fost rezolvata adaugand o conditie inainte de actualizare care verifica daca utilizatorul a selectat sau nu un elev.

if(studentSelector.getSelectedIndex() != -1) {  
 numeTF.setText(elevi.get(studentSelector.getSelectedIndex()).getNume());  
 prenumeTF.setText(elevi.get(studentSelector.getSelectedIndex()).getPrenume());  
 numarMatricolTF.setText(elevi.get(studentSelector.getSelectedIndex()).getId());  
 clasaTF.setText(classSelector.getSelectedItem().toString());  
}

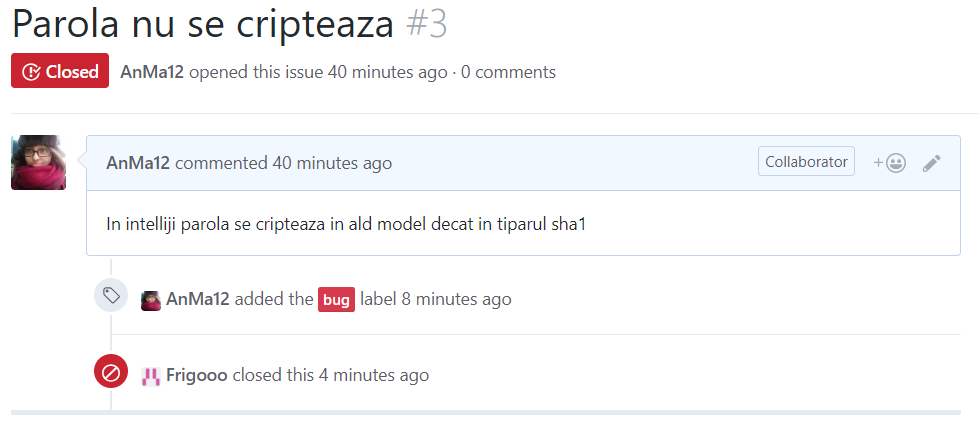


### Bug 3

# Parola nu se cripteaza

# <https://github.com/fr0yo1/CatalogOnline/issues/3>

public void updateParolaBD(String username, char[] parolaNoua) throws SQLException, ClassNotFoundException {  
 System.*out*.println("Updating parola");  
 stmt = conn.createStatement();  
 String sql = "UPDATE LoginData\n" +  
 "SET password = SHA1('" + new String(parolaNoua) +  
 "') WHERE username = '" + username + "';";  
 stmt.executeUpdate(sql);  
 System.*out*.println("Updated parola");  
}



### Bug 4

# Media generala la elevi bug

# <https://github.com/fr0yo1/CatalogOnline/issues/4>

if(note.size()==0)  
 return 0;  
return suma/note.size();

### 

# Refactoring

Cod initial pentru conexiunea/login Baza de date:

package com.company.database;  
  
import java.sql.\*;  
public class LoginDatabase {  
 // JDBC driver name and database URL  
 public static final String *JDBC\_DRIVER* = "com.mysql.jdbc.Driver";  
 public static final String *DB\_URL* = "jdbc:mysql://myownpi.ddns.net:3306/catalog";  
  
 // Database credentials  
 public static final String *USER* = "admin";  
 public static final String *PASS* = "admin";  
  
 public static Connection *conn* = null;  
 public static Statement *stmt* = null;  
  
 public static void createConnection() throws ClassNotFoundException, SQLException {  
 //STEP 2: Register JDBC driver  
 Class.*forName*("com.mysql.jdbc.Driver");  
  
 //STEP 3: Open a connection  
 System.*out*.println("Connecting to a selected database...");  
 *conn* = DriverManager.*getConnection*(*DB\_URL*, *USER*, *PASS*);  
 System.*out*.println("Connected database successfully...");  
 }  
}

Cod dupa refactoring pentru conexiunea/login Baza de date:

package loginDatabase;  
  
import java.sql.\*;  
public class LoginDatabase  
{ private Connection connection = null;  
 private String id;  
 private String statut;  
 private String nume;  
 private String prenume;  
 static private LoginDatabase *ad*;  
 private LoginDatabase(String user,char[] password){  
 try {  
 connection = DriverManager.*getConnection*(  
 "jdbc:mysql://myownpi.ddns.net:3306/register?allowMultiQueries=true", "admin", "admin"  
 );  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 PreparedStatement stmt=null;  
 try {  
 stmt = connection.prepareStatement("SELECT \* FROM LoginData WHERE username=? and password=SHA1(?)");  
 stmt.setString(1, user);  
 ((PreparedStatement) stmt).setString(2, new String(password));  
 ResultSet rs=stmt.executeQuery();  
 if(!rs.next()){  
 connection.close();  
 connection=null;  
 }  
 else  
 {  
 id=rs.getString(3);  
 statut=rs.getString(4);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 try {  
 String query= "";  
 Statement stmt1 = null;  
 stmt1 = connection.createStatement();  
 switch (statut) {  
 case "3" : query = "SELECT nume,prenume FROM Parinte WHERE id\_parinte = " + id + ";"; break;  
 case "2" : query = "SELECT nume,prenume FROM Elev WHERE id\_elev = " + id + ";"; break;  
 default : query = "SELECT nume,prenume FROM Profesor WHERE id\_profesor = " + id + ";"; break;  
 }  
 System.*out*.println(query);  
 ResultSet rs=stmt1.executeQuery(query);  
 if(rs.next())  
 {   
 nume = rs.getString("nume");  
 prenume =rs.getString("prenume");  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
   
   
 }  
 static public LoginDatabase getAccess(String user,char[] password){  
 if(*ad*!=null)  
 if(*ad*.connection!=null)  
 return *ad*;  
  
 *ad*=new LoginDatabase(user,password);  
 return *ad*;  
 }  
 public String getId()  
 {  
 return id;  
 }  
 public Connection getConnection()  
 {  
 return connection;  
 }  
 public String getStatut()  
 {  
 return statut;  
 }  
}

Refactoring-ul a fost facut deoarece initial clasa LoginDatabase era statica, urmand mai apoi ca si metoda createConnection era un statica si publica. Dupa refactoring, clasa LoginDatabase nu mai este static, iar metodele din ea pentru realizarea conexiunii sunt private. Astfel, se respecta principiul incapsularii iar asta ne-a ajutat sa avem un control mult mai ridicat asupra conexiunii.

# Design patterns

### Singleton

private LoginDatabase(String user,char[] password){  
 try {  
 connection = DriverManager.*getConnection*(  
 "jdbc:mysql://myownpi.ddns.net:3306/register?allowMultiQueries=true", "admin", "admin"  
 );  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 PreparedStatement stmt=null;  
 try {  
 stmt = connection.prepareStatement("SELECT \* FROM LoginData WHERE username=? and password=SHA1(?)");  
 stmt.setString(1, user);  
 ((PreparedStatement) stmt).setString(2, new String(password));  
 ResultSet rs=stmt.executeQuery();  
 if(!rs.next()){  
 connection.close();  
 connection=null;  
 }  
 else  
 {  
 id=rs.getString(3);  
 statut=rs.getString(4);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 try {  
 String query= "";  
 Statement stmt1 = null;  
 stmt1 = connection.createStatement();  
 switch (statut) {  
 case "3" : query = "SELECT nume,prenume FROM Parinte WHERE id\_parinte = " + id + ";"; break;  
 case "2" : query = "SELECT nume,prenume FROM Elev WHERE id\_elev = " + id + ";"; break;  
 default : query = "SELECT nume,prenume FROM Profesor WHERE id\_profesor = " + id + ";"; break;  
 }  
 System.*out*.println(query);  
 ResultSet rs=stmt1.executeQuery(query);  
 if(rs.next())  
 {   
 nume = rs.getString("nume");  
 prenume =rs.getString("prenume");  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
   
   
 }  
 static public LoginDatabase getAccess(String user,char[] password){  
 if(*ad*!=null)  
 if(*ad*.connection!=null)  
 return *ad*;  
  
 *ad*=new LoginDatabase(user,password);  
 return *ad*;  
 }

Conexiunea la baza de date se obtine folosind clasa singleton LoginDatabse. Scopul acestei abordari este de a asigura unicitatea conexiunii, astfel un utilizator poate obtine o singura conexiune la un moment dat.