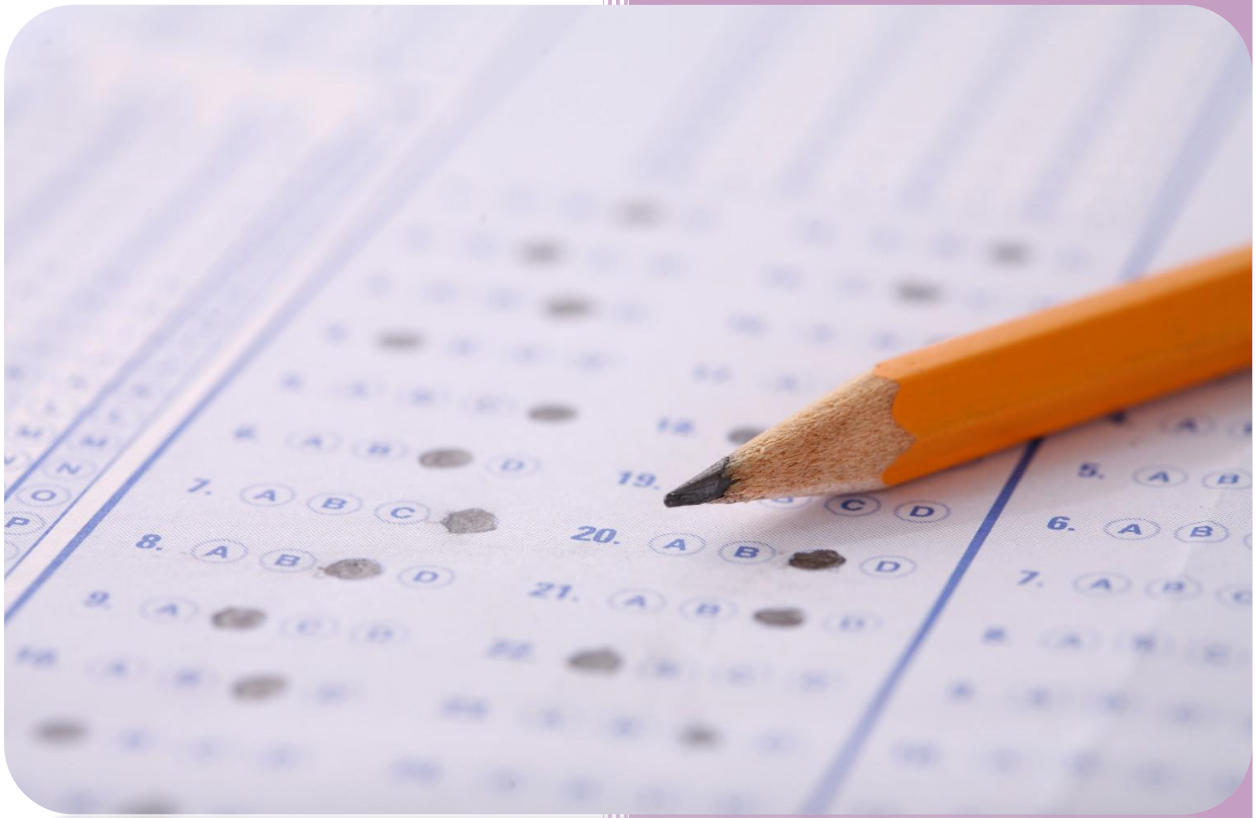
2015

Newdle System – Final Report



Juan Pablo Rodríguez Valentín

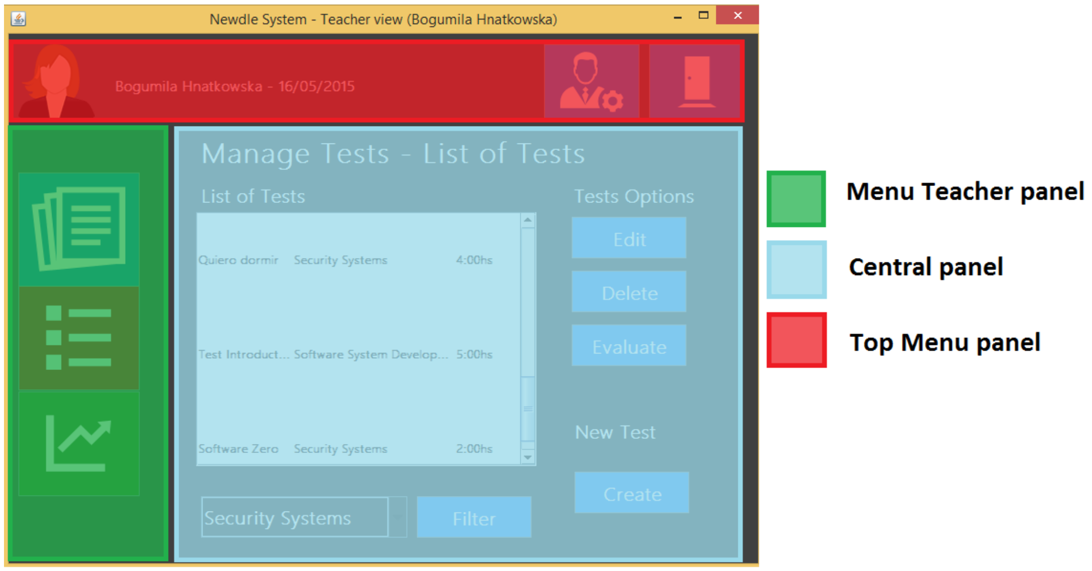
25-6-2015

Table of Contents

Interface and functionality

Interface Distribution

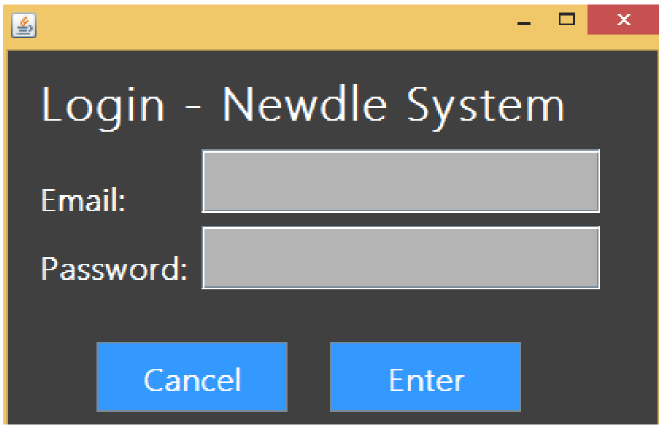
As we can see in the interface, is possible to identify three different panels in which the information is divided:


Newdle System - Teacher view (Bogumila Hnatkowska)
Bogumila Hnatkowska - 16/05/2015
Manage Tests - List of Tests
List of Tests
Tests Options
Quiero dormir Security Systems 4:00hs
Test Introduct... Software System Develop... 5:00hs
Software Zero Security Systems 2:00hs
Edit
Delete
Evaluate
New Test
Create
Security Systems
Filter
🟩 Menu Teacher panel
🟦 Central panel
🟥 Top Menu panel

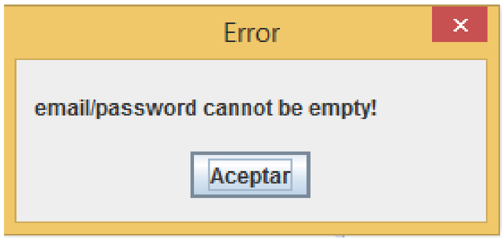
The application is going to follow this distribution for all the users, in order to achieve the non-functional requirement "Usability".

Login

In order to manage the Access and achieve the non-functional requirement "Security", the first interface to access to the application is a login View, manage by Controller Login which control and log the different attempts and errors. A deeply explanation about the algorithm is located in section 1.A – Architectural Mechanisms

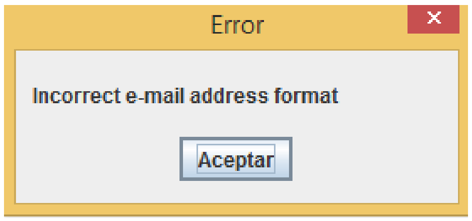


Right information:
User email: teacher@gmail.com
Password: HOla00==

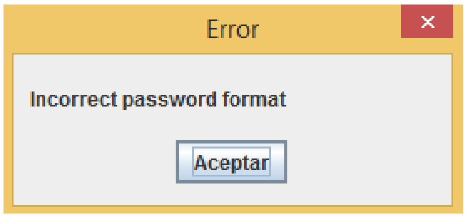


email/password cannot be empty!

Empty inputs error.

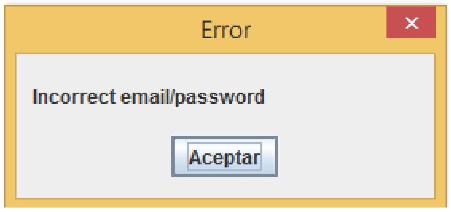


Incorrect e-mail address format



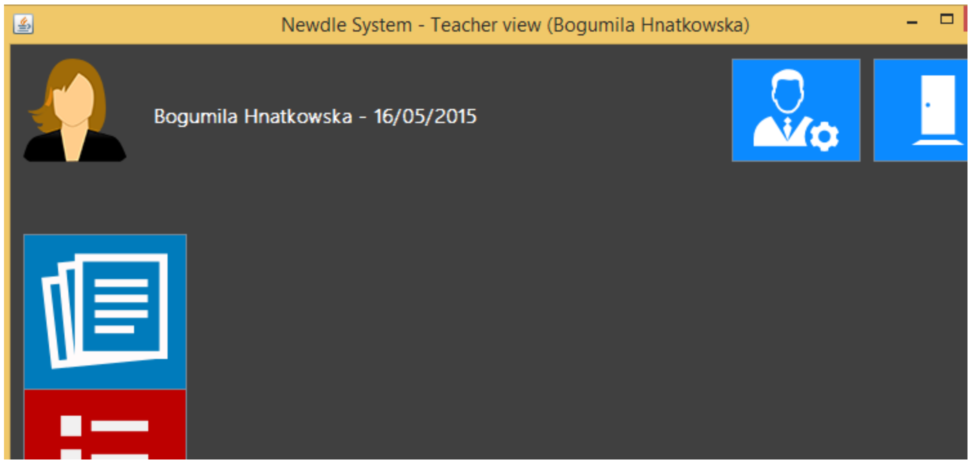
Incorrect password format

Wrong email/password format errors.



Incorrect email/password

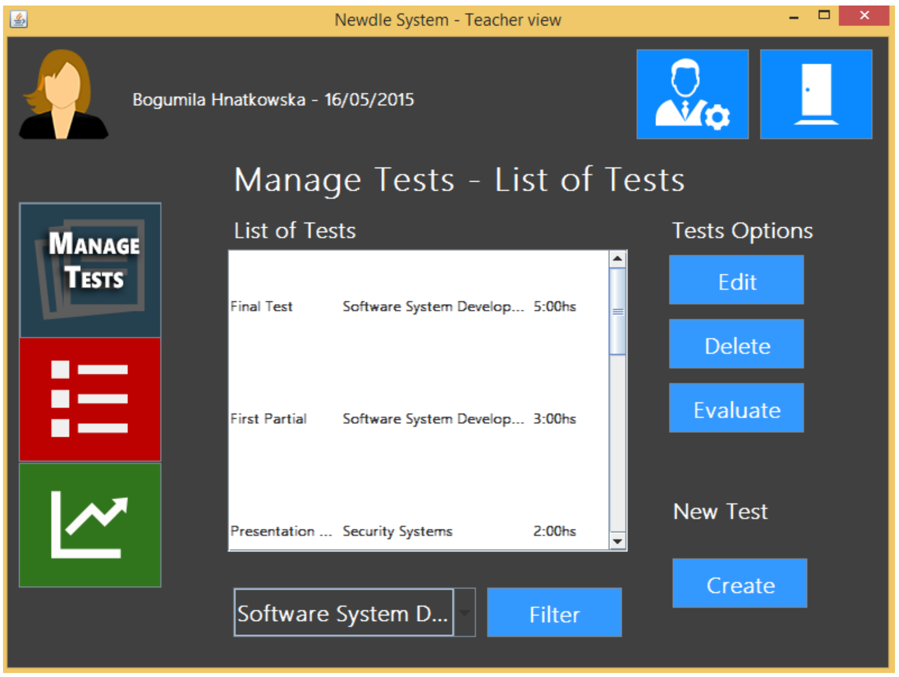
Wrong email/password error.


Newdle System - Teacher view (Bogumila Hnatkowska)
Bogumila Hnatkowska - 16/05/2015

Teacher interface – Welcome Page

List tests

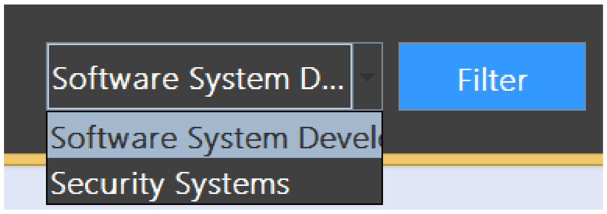
When the teacher click the button "Manage Test", the controller load in the central panel the list of the tests that belong to all the courses. Also the different options about tests:



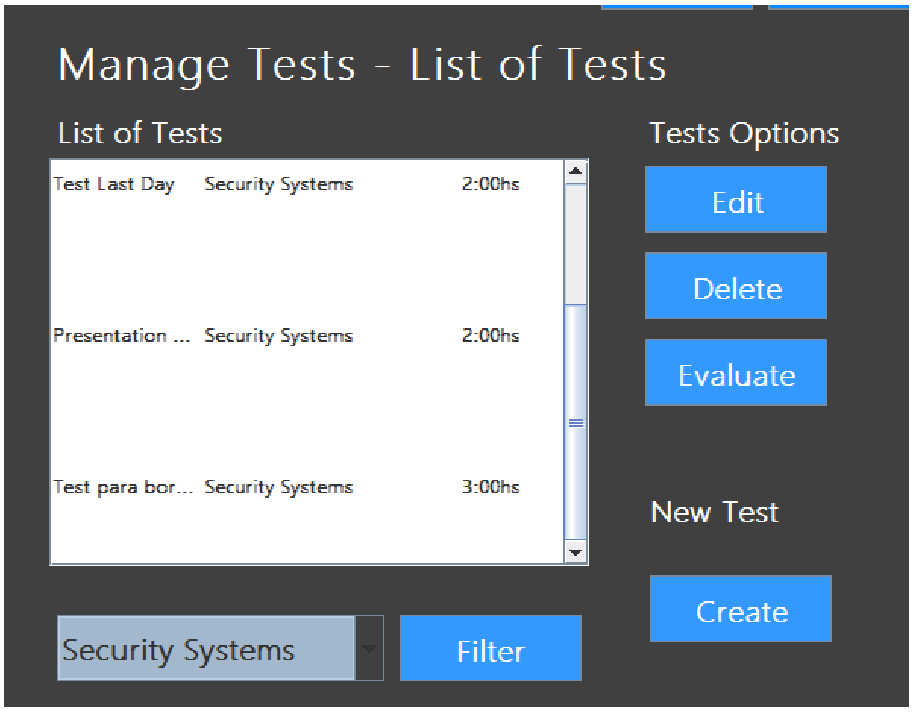
Final Test	Software System Develop...	5:00hs
First Partial	Software System Develop...	3:00hs
Presentation ...	Security Systems	2:00hs

Filter tests by Subject:

The application allow also filter by subject, in order to help to the teacher to find test easily:



User should choose the name of the course and click the button.

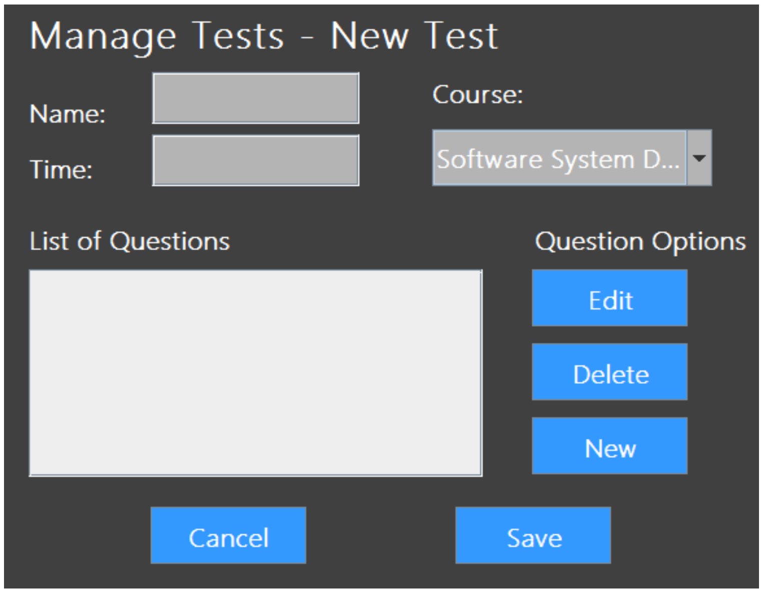


Test Last Day	Security Systems	2:00hs
Presentation ...	Security Systems	2:00hs
Test para bor...	Security Systems	3:00hs

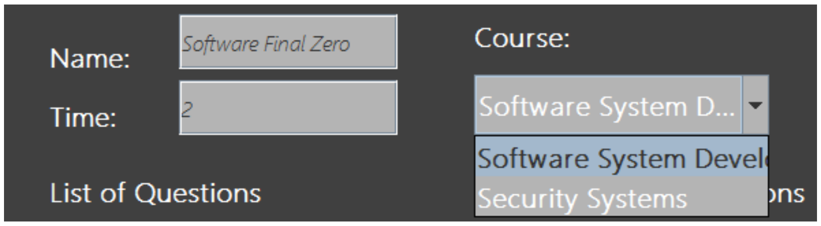
The controller refresh the view in order to show just the test which belongs to the selected course.

Create a test

When the teacher click the button create, the controller load in the central panel:

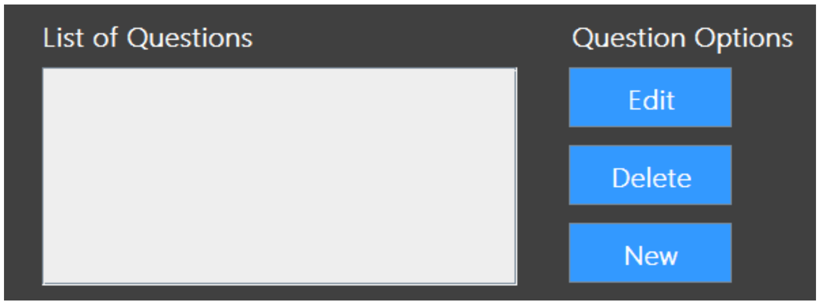


As we can see, the panel is divided in two different parts: test info and questions:

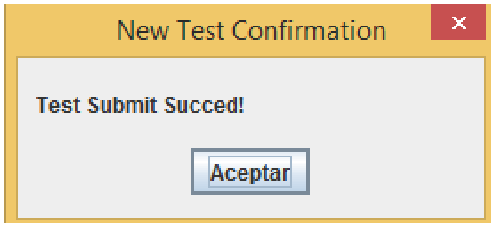


Name: Software Final Zero
Time: 2

In test info, teacher could introduce all the basic information about new test, like Name, Time and Course.



In questions, teacher could manage all the questions that belong to the test. In this case, there isn't any question because is a new test. In order to add a new one, is necessary to follow the step 3.1

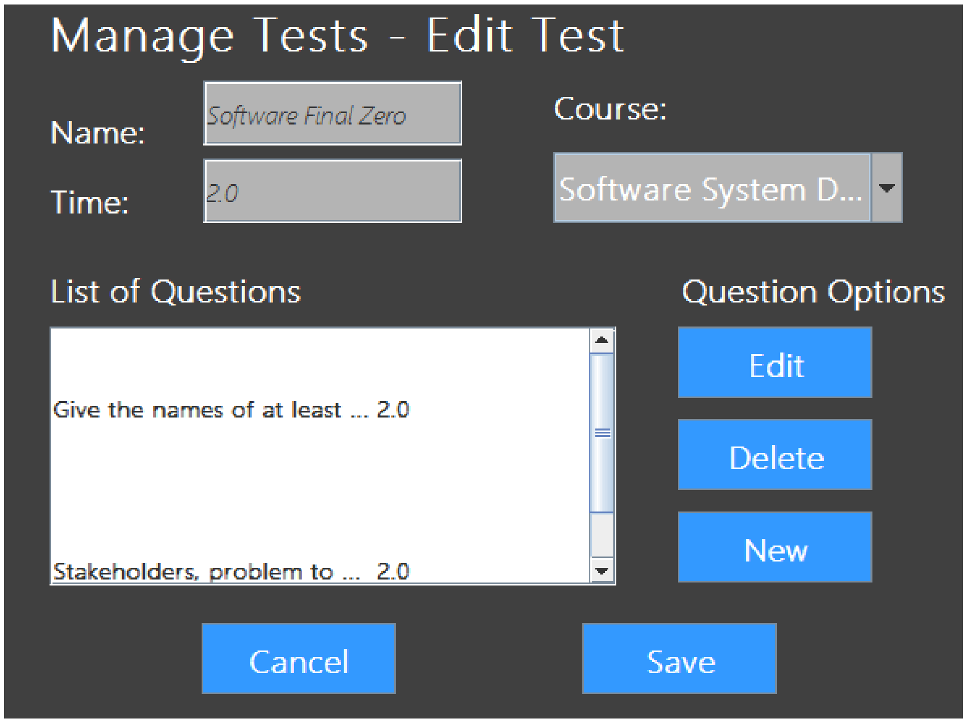


Test Submit Succed!

After fill up the inputs and click the button save, the application confirm the operation.

Edit test

Similar to create test, when the teacher select a test and click the button "Edit", the controller load in the central panel the "Edit Test" view:



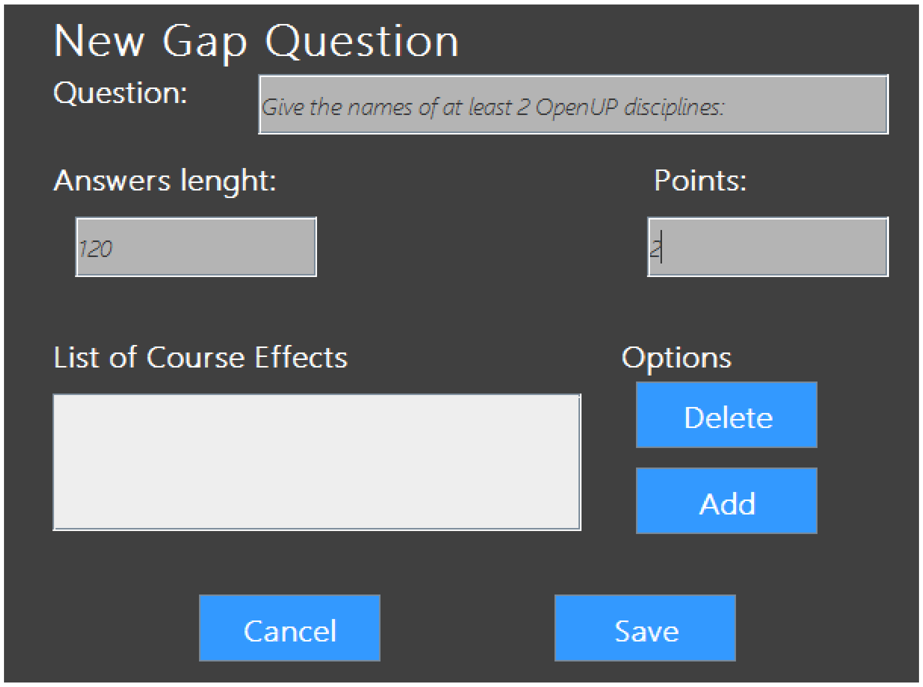
Name: Software Final Zero
Time: 2.0
Course: Software System D...

Give the names of at least ... 2.0
Stakeholders, problem to ... 2.0

Here is possible to change the Name, time or course and also add new questions.

Create a gap question and add to test

When the teacher click in new question, and choose "Gap question", Controller load the "New gap question" view in the central panel.

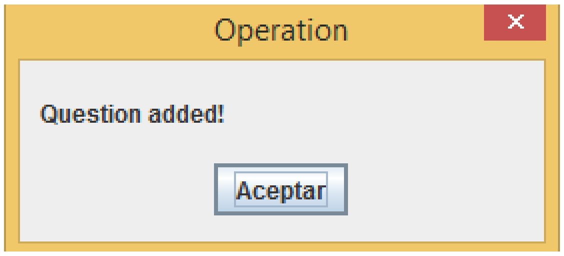


Give the names of at least 2 OpenUP disciplines:

120

2

Following the same distribution that new test view, there are 2 different parts, one with the basic information about the question, and the other one which allow to manage the Course Effects connected with the question.

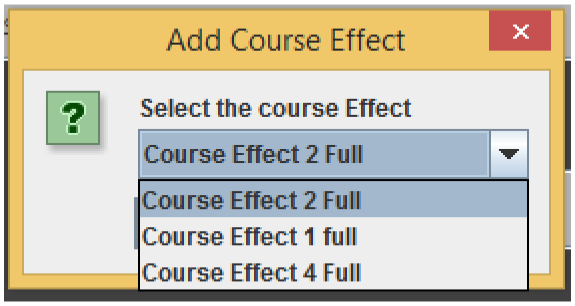


Question added!

After fill up all the information and click the button "Save", Controller advice to the teacher about the operation, and load the "Edit test" view, with the updated information.

Add Course effects to question

As we mention before, is possible to connect a question with the Course Effects.



Select the course Effect

Course Effect 2 Full

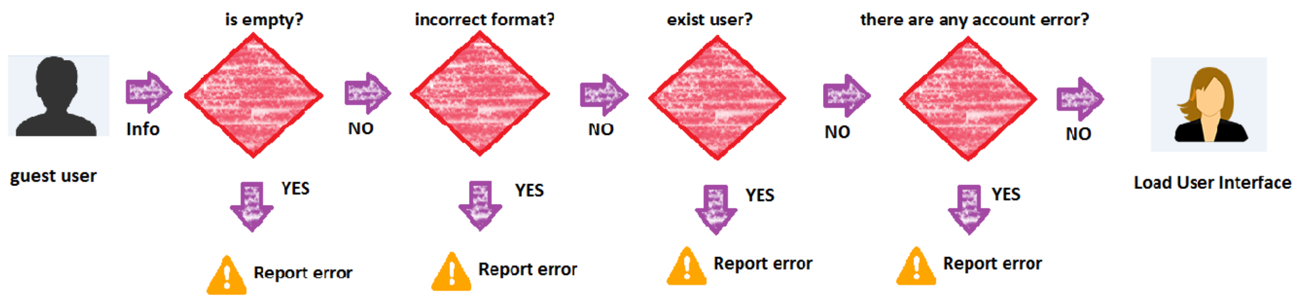
Course Effect 2 Full
Course Effect 1 full
Course Effect 4 Full

Architectural Mechanisms

1A. Login using at least a email and a password

Is forbidden the access to any of the restricted user interface without be registered in the system. There are a special and limited user interface to unlogged users which just allow to fill up a form in order to sign in.

In order to fulfil this architectural Mechanism, the Login Controller follow next algorithm:

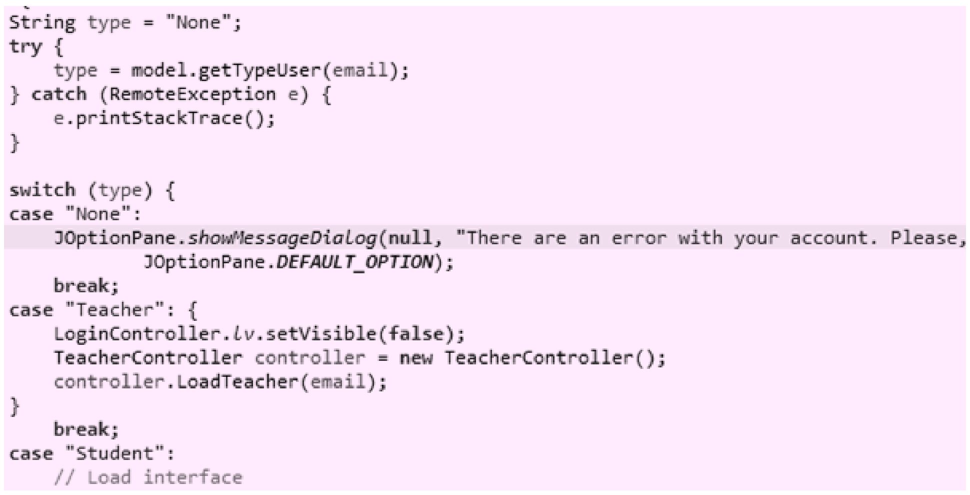


User introduce email/password in Login View, and controller control all the possible mistakes/errors. At the end, if the information is correct, the controller check the kind of user and create a new instance of the proper interface (In this iteration, just exist Teacher interface).

1B. Different user interfaces

According to their profile, users are going to access to different Interfaces with different user menus and functionalities.

As we mention in the 1A section, Login controller is able to identify the user type in order to create a proper user interface:



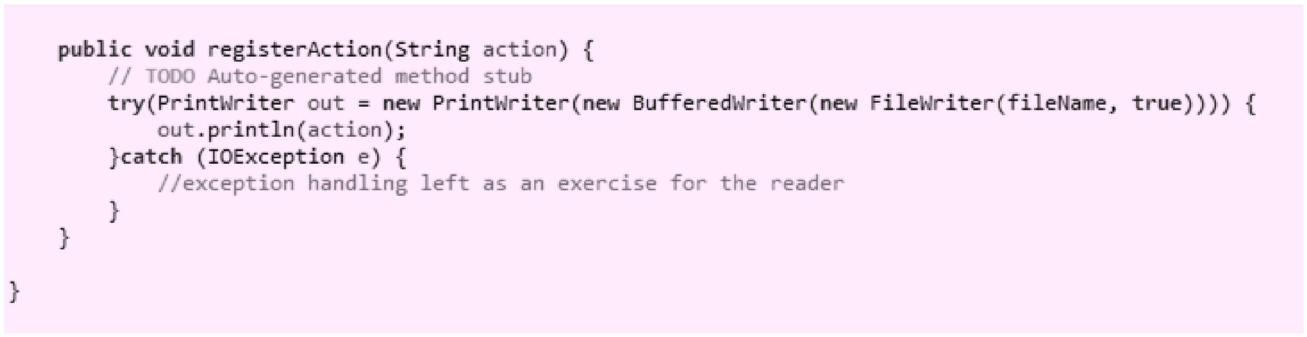
```
String type = "None";
try {
    type = model.getTypeUser(email);
} catch (RemoteException e) {
    e.printStackTrace();
}

switch (type) {
case "None":
    JOptionPane.showMessageDialog(null, "There are an error with your account. Please,
            JOptionPane.DEFAULT_OPTION);
    break;
case "Teacher": {
    LoginController.lv.setVisible(false);
    TeacherController controller = new TeacherController();
    controller.LoadTeacher(email);
}
    break;
case "Student":
    // Load interface
```

Controller/LoginController.java

1C. Auditing: Every sensitive action can be logged

Every action performed by users is going to be register in a log file in the server, with information like dates, hour and IP address.

In order to fulfil this architectural Mechanism, the Controller create a log using LogRegister class, (package utils), using next code:



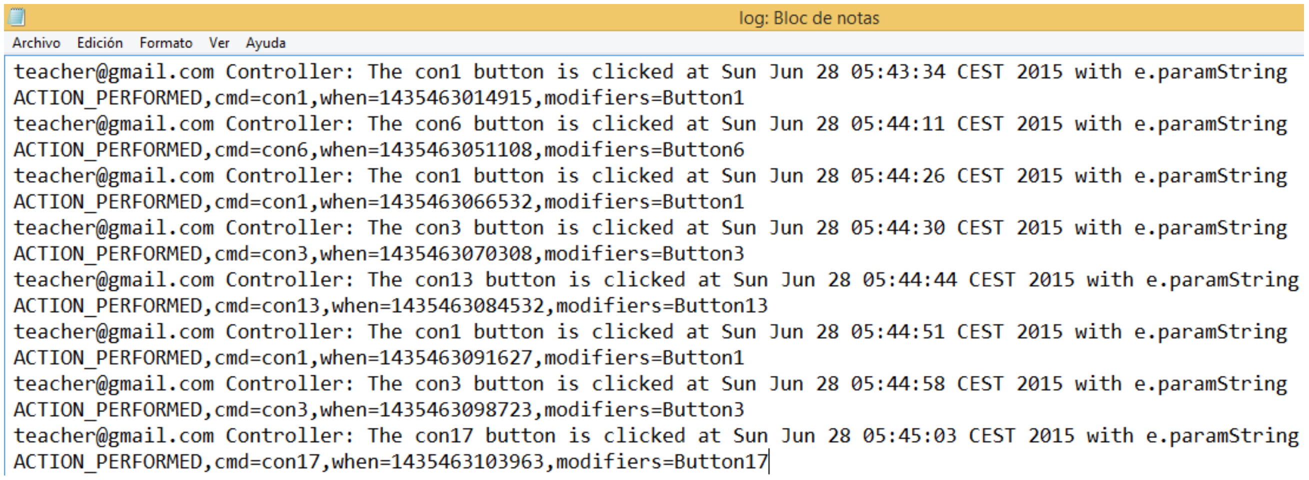
```
    public void registerAction(String action) {
        // TODO Auto-generated method stub
        try(PrintWriter out = new PrintWriter(new BufferedWriter(new FileWriter(fileName, true)))) {
            out.println(action);
        }catch (IOException e) {
            //exception handling left as an exercise for the reader
        }
    }

}
```

utils/LogRegister.java

Every interaction of every user is register using the format:

Email+ “text” + action code + “text” + date+ “text” + parameters

Example:



```
teacher@gmail.com Controller: The con1 button is clicked at Sun Jun 28 05:43:34 CEST 2015 with e.paramString
ACTION_PERFORMED,cmd=con1,when=1435463014915,modifiers=Button1
teacher@gmail.com Controller: The con6 button is clicked at Sun Jun 28 05:44:11 CEST 2015 with e.paramString
ACTION_PERFORMED,cmd=con6,when=1435463051108,modifiers=Button6
teacher@gmail.com Controller: The con1 button is clicked at Sun Jun 28 05:44:26 CEST 2015 with e.paramString
ACTION_PERFORMED,cmd=con1,when=1435463066532,modifiers=Button1
teacher@gmail.com Controller: The con3 button is clicked at Sun Jun 28 05:44:30 CEST 2015 with e.paramString
ACTION_PERFORMED,cmd=con3,when=1435463070308,modifiers=Button3
teacher@gmail.com Controller: The con13 button is clicked at Sun Jun 28 05:44:44 CEST 2015 with e.paramString
ACTION_PERFORMED,cmd=con13,when=1435463084532,modifiers=Button13
teacher@gmail.com Controller: The con1 button is clicked at Sun Jun 28 05:44:51 CEST 2015 with e.paramString
ACTION_PERFORMED,cmd=con1,when=1435463091627,modifiers=Button1
teacher@gmail.com Controller: The con3 button is clicked at Sun Jun 28 05:44:58 CEST 2015 with e.paramString
ACTION_PERFORMED,cmd=con3,when=1435463098723,modifiers=Button3
teacher@gmail.com Controller: The con17 button is clicked at Sun Jun 28 05:45:03 CEST 2015 with e.paramString
ACTION_PERFORMED,cmd=con17,when=1435463103963,modifiers=Button17
```

2A. System is going to allow to export the data in CVS format

Following the example of Edukacja template, using sematic rules and strategies the system will be able to export the information in a compatible format with Edukacja.

This architectural mechanism will be fulfil in upcoming iterations.

3A. Data persistence will be addressed using a relational database

Following the architecture described in the Development View, the system will be use DAO (Data access object) to mapping the application calls to the persistence layer and provide some specific data operations without exposing details of the database.

As you can see in the 3.1 section, the application use the DAO pattern in order to access and manage the information of database.

3B. Periodical backups of the database

In order to preserve the data in case of equipment failure or other catastrophe, the system will be perform a full back up every week at midnight of Sunday, and an incremental backup every day at 2AM.

This architectural mechanism will be fulfil in upcoming iterations.

4A. Maintenance activities and backups.

The System must be availability at least 16/7, the time left (8 hours) is reserved for any maintenance activities and backups.

This architectural mechanism will be fulfil in upcoming iterations.

4B. Database is going to be localized in the Wrocław University of Technology's Server.

Wrocław University of Technology's Server have a strict policy which allow to ensure security and availability to users.

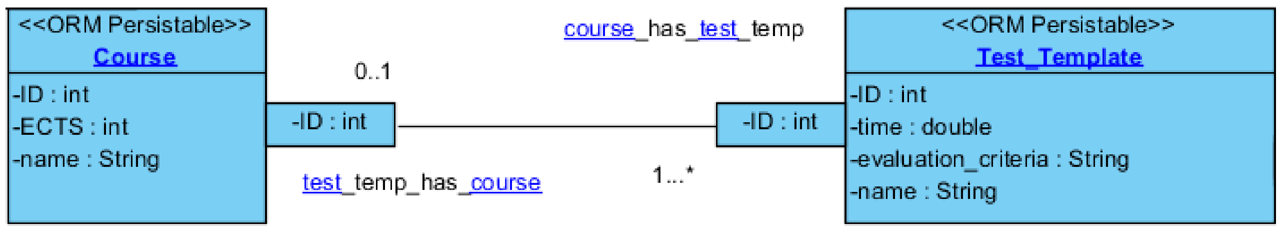
This architectural mechanism will be fulfil in upcoming iterations, because in that iteration the database is located in a local server.

3 Others

3.1 Process from Visual Paradigm to Code

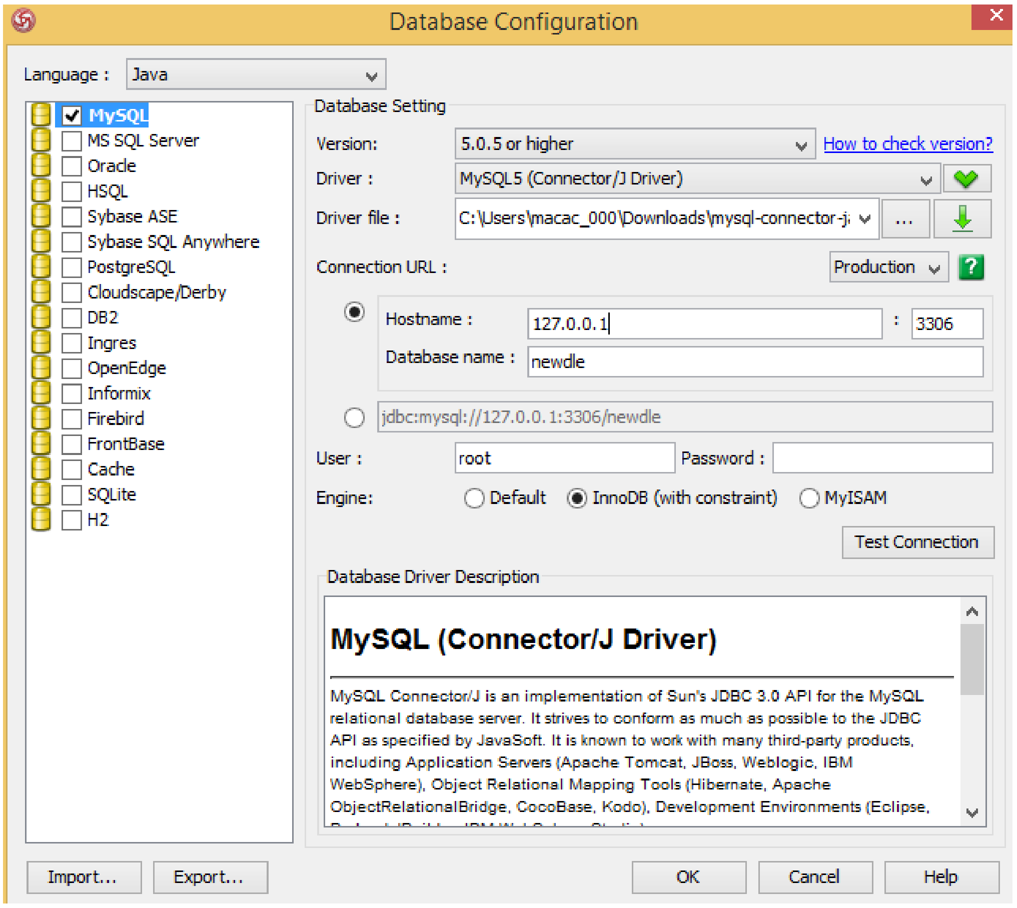
As we have studied during the subject, we could use Visual Paradigm for UML in order to generate the Physical model of our data from the Conceptual model (class diagram). Is possible also to generate partially the source code to manage the data-access of our application following the DAO pattern.

The different steps followed to use this functionality are next
1) is necessary assign to every class that represent a class the stereotype <<ORM Persistable>>. Also is necessary to assign the multiplicity, rol name and Qualifier to the relations, as we can see in next example:



Basic example of class diagram adapted to code-generation in VP4UML.

2) Now is necessary to connect our Project with a Database server. In our case is MySQL server running in a local server:

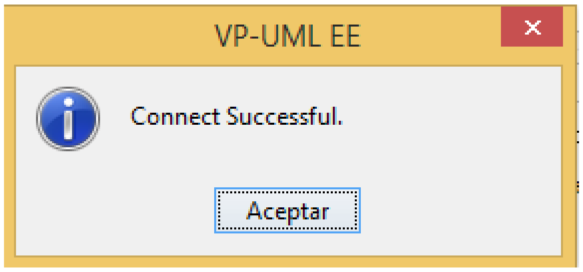


Version: 5.0.5 or higher
Driver : MySQL5 (Connector/J Driver)
Driver file : C:\Users\macac_000\Downloads\mysql-connector-j
Connection URL : Production
Hostname : 127.0.0.1 : 3306
Database name : newdle
jdbc:mysql://127.0.0.1:3306/newdle
User : root Password :
Engine: Default [✓ InnoDB (with constraint)] MyISAM

MySQL (Connector/J Driver)

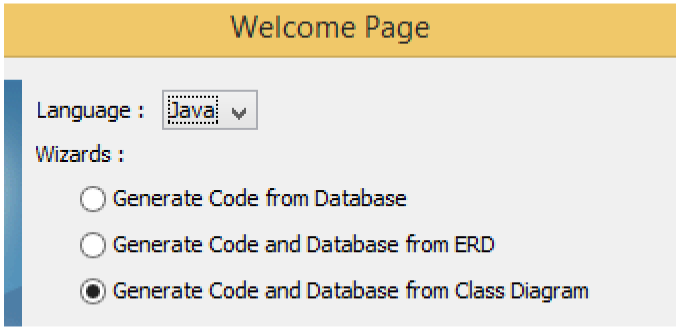
MySQL Connector/J is an implementation of Sun's JDBC 3.0 API for the MySQL relational database server. It strives to conform as much as possible to the JDBC API as specified by JavaSoft. It is known to work with many third-party products, including Application Servers (Apache Tomcat, JBoss, Weblogic, IBM WebSphere), Object Relational Mapping Tools (Hibernate, Apache ObjectRelationalBridge, CocoBase, Kodo), Development Environments (Eclipse,

Also is necessary to create a database in the server, and configure the parameters. We could test if everything is right click the "Test Connection" button:



Connect Successful.

3) After select ORM/Wizards, we could open different Wizard in order to generate Code in different languages (Java) from different diagrams, in our case, Class Diagram:



Welcome Page

Language : Java

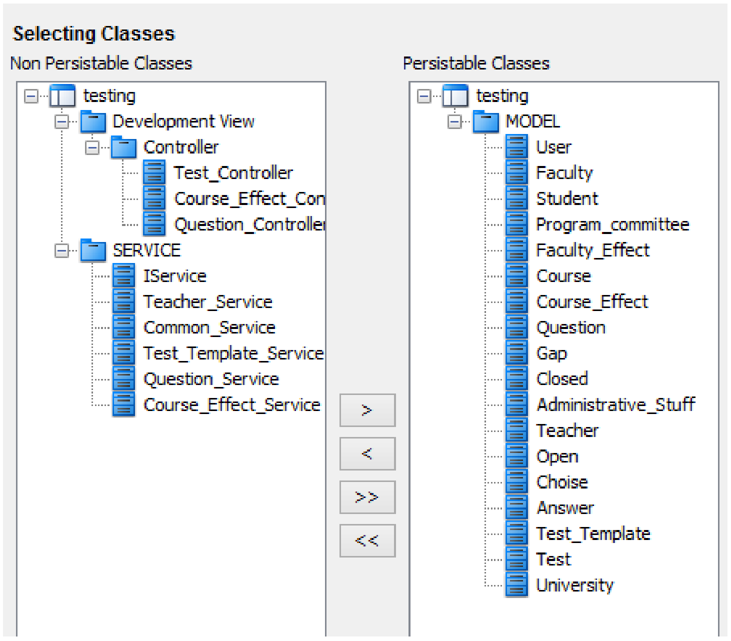
Wizards :

( ) Generate Code from Database

( ) Generate Code and Database from ERD

(●) Generate Code and Database from Class Diagram

3.1) Wizard start offering the possibility to choose the classes that represent our database tables:



Selecting Classes

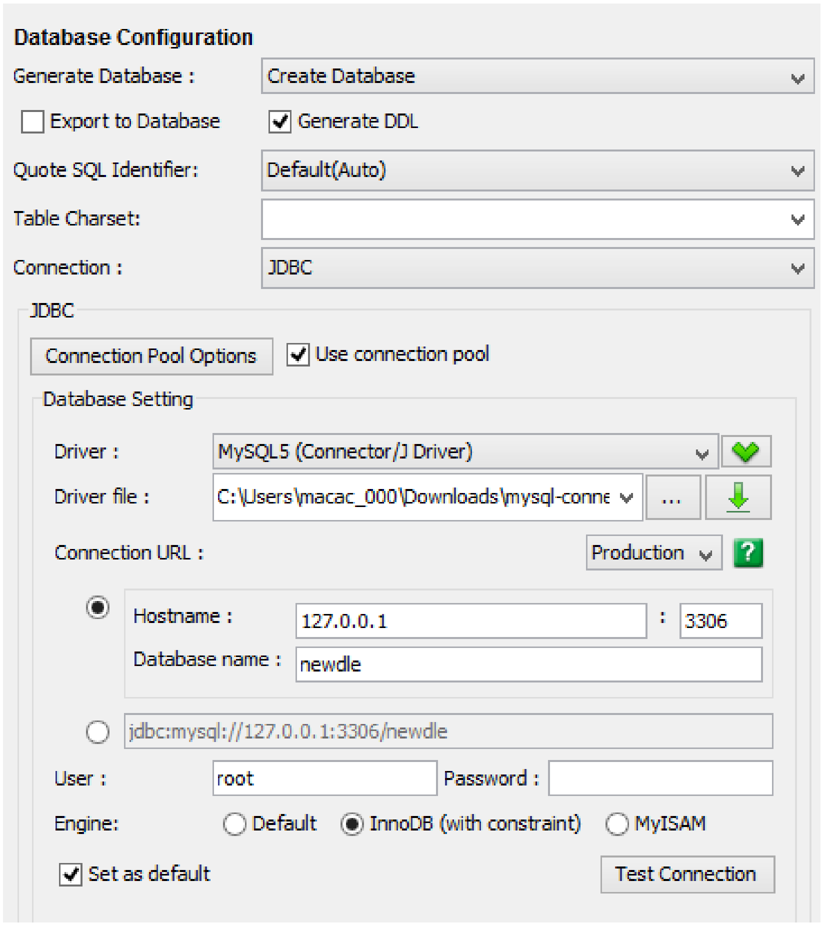
Non Persistable Classes

- testing
  - Development View
    - Controller
      - Test_Controller
      - Course_Effect_Con
      - Question_Controlle
  - SERVICE
    - IService
    - Teacher_Service
    - Common_Service
    - Test_Template_Service
    - Question_Service
    - Course_Effect_Service

Persistable Classes

- testing
  - MODEL
    - User
    - Faculty
    - Student
    - Program_committee
    - Faculty_Effect
    - Course
    - Course_Effect
    - Question
    - Gap
    - Closed
    - Administrative_Stuff
    - Teacher
    - Open
    - Choise
    - Answer
    - Test_Template
    - Test
    - University

Next is possible to choose the primary keys and other parameters. After click next button, is possible to change the database parameters in case that we need:



Database Configuration

Generate Database : Create Database

[ ] Export to Database [✓] Generate DDL

Quote SQL Identifier: Default(Auto)

Table Charset:

Connection : JDBC

JDBC

Connection Pool Options [✓] Use connection pool

Database Setting

Driver : MySQL5 (Connector/J Driver)

Driver file : C:\Users\macac_000\Downloads\mysql-conne

Connection URL : Production

(●) Hostname : 127.0.0.1 : 3306

Database name : newdle

( ) jdbc:mysql://127.0.0.1:3306/newdle

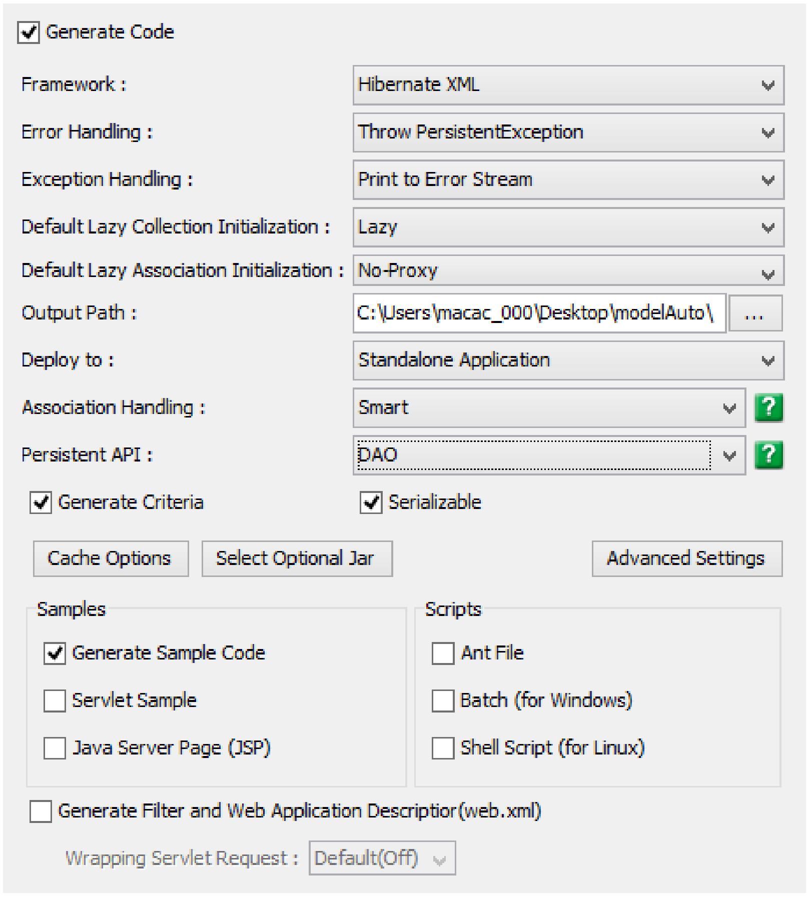
User : root Password :

Engine: ( ) Default (●) InnoDB (with constraint) ( ) MyISAM

[✓] Set as default

Test Connection

Last is the most important Step. We need to choose the Persistent API (DAO in our case), and select the destination folder, among other parameters:



☑ Generate Code

Framework :	Hibernate XML
Error Handling :	Throw PersistentException
Exception Handling :	Print to Error Stream
Default Lazy Collection Initialization :	Lazy
Default Lazy Association Initialization :	No-Proxy
Output Path :	C:\Users\macac_000\Desktop\modelAuto\
Deploy to :	Standalone Application
Association Handling :	Smart
Persistent API :	DAO

☑ Generate Criteria ☑ Serializable

Samples
- ☑ Generate Sample Code
- ☐ Servlet Sample
- ☐ Java Server Page (JSP)

Scripts
- ☐ Ant File
- ☐ Batch (for Windows)
- ☐ Shell Script (for Linux)

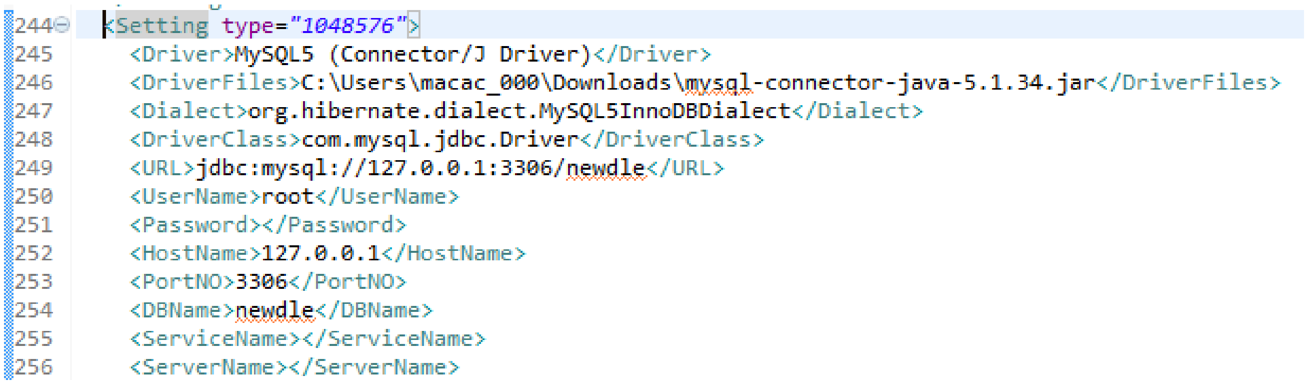
☐ Generate Filter and Web Application Descriptor(web.xml)

Wrapping Servlet Request : Default(Off)

If we follow the steps correctly, code and ERD diagram is generated, and also the DDL file to create the database. Also is necessary to attach the orm.jar library to our project in order to use the code generated.

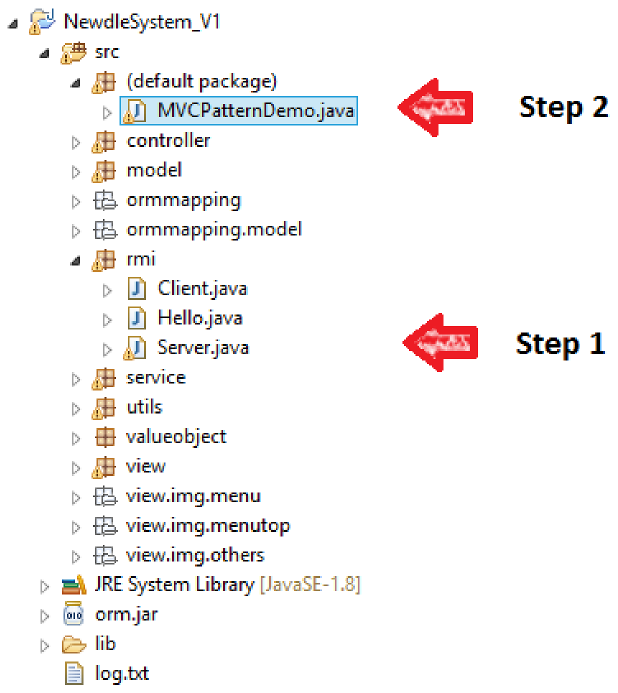
3.2 Access to source codes

There are a special folder in Dropbox called source code with a ZIP file with all the source code and also a file called testing.ddl to create the newdle database in the server. In case that we need to change some parameters, like IP address or port, the configuration data file is located in "ormmapping/database.cfg.xml".



```xml
<Setting type="1048576">
    <Driver>MySQL5 (Connector/J Driver)</Driver>
    <DriverFiles>C:\Users\macac_000\Downloads\mysql-connector-java-5.1.34.jar</DriverFiles>
    <Dialect>org.hibernate.dialect.MySQL5InnoDBDialect</Dialect>
    <DriverClass>com.mysql.jdbc.Driver</DriverClass>
    <URL>jdbc:mysql://127.0.0.1:3306/newdle</URL>
    <UserName>root</UserName>
    <Password></Password>
    <HostName>127.0.0.1</HostName>
    <PortNO>3306</PortNO>
    <DBName>newdle</DBName>
    <ServiceName></ServiceName>
    <ServerName></ServerName>
```

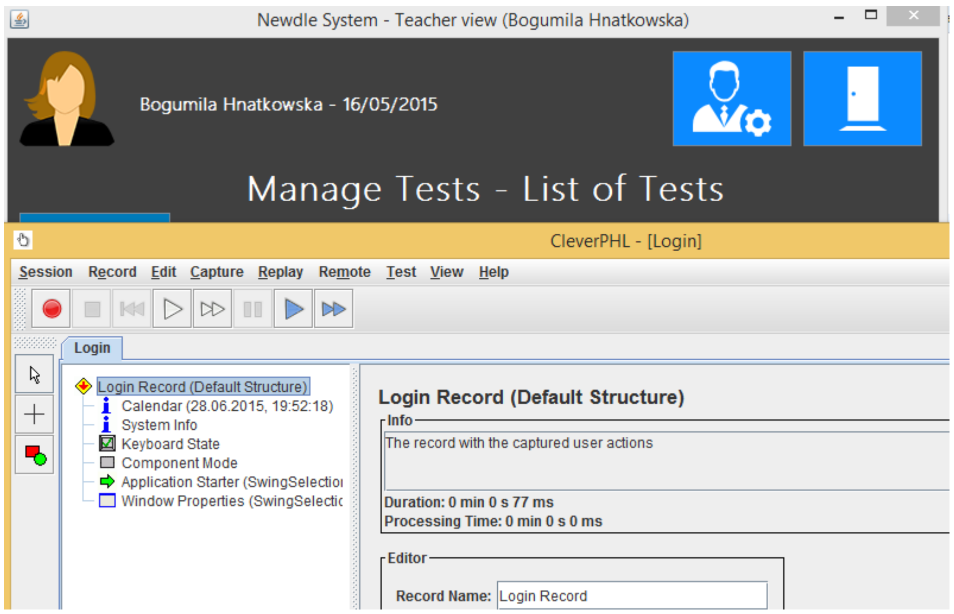
The right way to made it work after configure the database and import the ORM library, is execute Server (rmi/server.java) in order to start the server, and after that Run MVCPatternDemo.java (MVCPatternDemo.java).



```
NewdleSystem_V1
    src
        (default package)
            MVCPatternDemo.java
        controller
        model
        ormmapping
        ormmapping.model
        rmi
            Client.java
            Hello.java
            Server.java
        service
        utils
        valueobject
        view
        view.img.menu
        view.img.menutop
        view.img.others
    JRE System Library [JavaSE-1.8]
    orm.jar
    lib
    log.txt
```

3.3 Software Testing

In order to test the application we use Jacareto, a software tool developed in Java which allow us to capture & replay the user interaction. Jacareto can be used for GUI tests, the creation of animated demonstrations and analyses of user behaviour [link].


Newdle System - Teacher view (Bogumila Hnatkowska)
Bogumila Hnatkowska - 16/05/2015
Manage Tests - List of Tests
CleverPHL - [Login]
Session Record Edit Capture Replay Remote Test View Help
Login

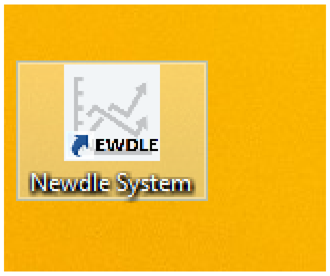
Login Record (Default Structure)
Calendar (28.06.2015, 19:52:18)
System Info
Keyboard State
Component Mode
Application Starter (SwingSelectio
Window Properties (SwingSelectic

Login Record (Default Structure)

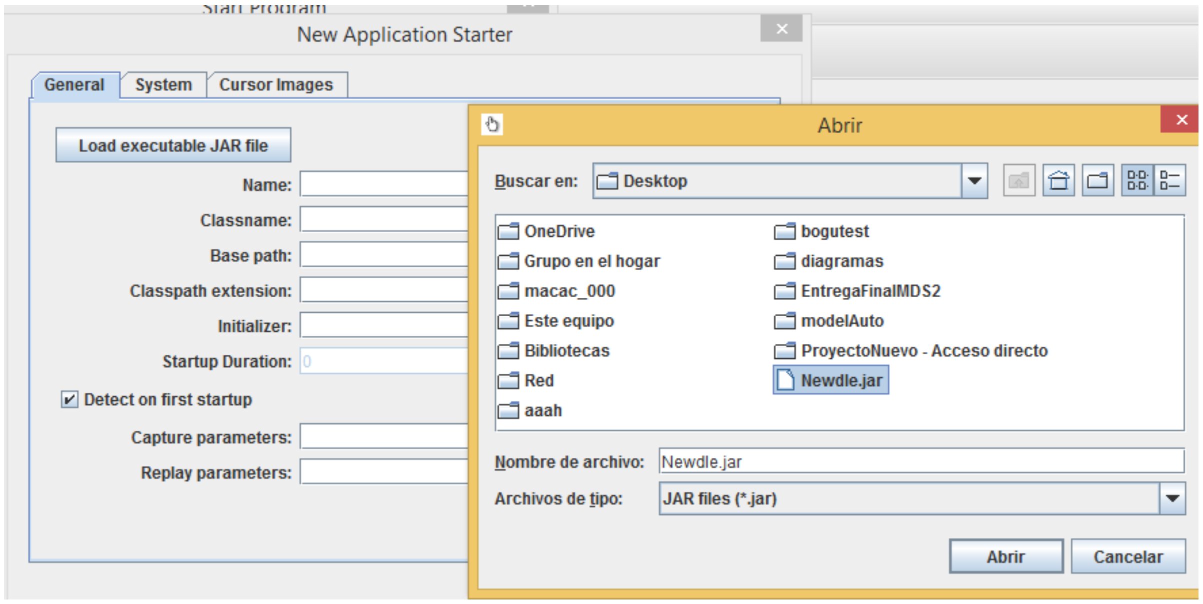
Info
The record with the captured user actions
Duration: 0 min 0 s 77 ms
Processing Time: 0 min 0 s 0 ms

Editor
Record Name: Login Record

In order to use this software, was necessary to export our Java application and create and executable jar file:



Now, we need to choose our application in CleverPHL:



New Application Starter

General | System | Cursor Images

Load executable JAR file

Name:
Classname:
Base path:
Classpath extension:
Initializer:
Startup Duration: 0
☑ Detect on first startup
Capture parameters:
Replay parameters:

Abrir

Buscar en: Desktop

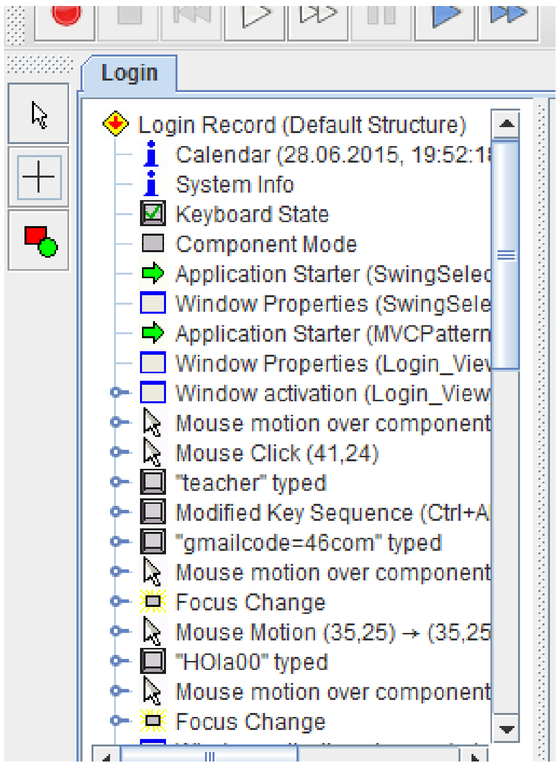
OneDrive	bogutest
Grupo en el hogar	diagramas
macac_000	EntregaFinalMDS2
Este equipo	modelAuto
Bibliotecas	ProyectoNuevo - Acceso directo
Red	Newdle.jar
aaah	

Nombre de archivo: Newdle.jar
Archivos de tipo: JAR files (*.jar)

Abrir | Cancelar

Session/StartProgram/New Application Starter – Load executable jar

Then, we need to select record and made all the input changes, and after that stop the record:

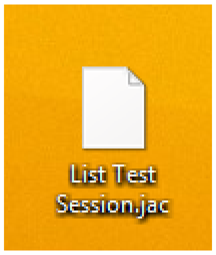


Login

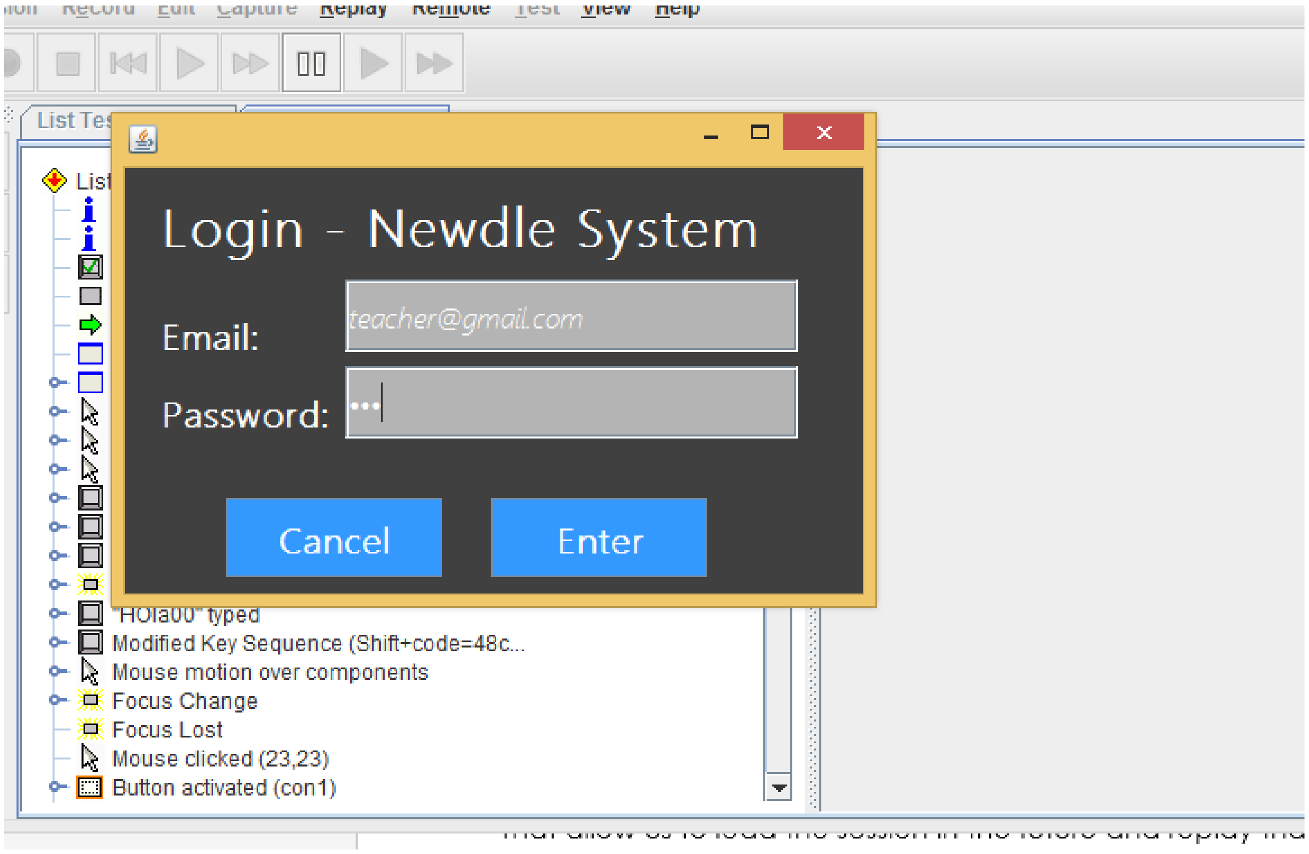
- Login Record (Default Structure)
  - Calendar (28.06.2015, 19:52:1
  - System Info
  - Keyboard State
  - Component Mode
  - Application Starter (SwingSelec
  - Window Properties (SwingSele
  - Application Starter (MVCPattern
  - Window Properties (Login_View
  - Window activation (Login_View
  - Mouse motion over component
  - Mouse Click (41,24)
  - "teacher" typed
  - Modified Key Sequence (Ctrl+A
  - "gmailcode=46com" typed
  - Mouse motion over component
  - Focus Change
  - Mouse Motion (35,25) → (35,25
  - "HOla00" typed
  - Mouse motion over component
  - Focus Change

All the interaction with the application were record.

After that we can replay the action, and also save the steps in .jac format:



That allow us to load the session in the future and replay that:



"HOla00" typed
Modified Key Sequence (Shift+code=48c...
Mouse motion over components
Focus Change
Focus Lost
Mouse clicked (23,23)
Button activated (con1)

For time reasons was not possible to include more than one test to this report: source-code/test/ List Test Session.jac.