EPITA Graduation School Of Computer Science & Advance Technologies

Java Fundamental Project

I AM CORE BISMARK Project

By bismark atta frimpong

Table of Contents

[Subject Description 4](#_Toc507275141)

[Subject analysis 4](#_Toc507275142)

[Major features 4](#_Toc507275143)

[Application Feasibility 5](#_Toc507275144)

[Data description 5](#_Toc507275145)

[Expected results 6](#_Toc507275146)

[Scope of the application 7](#_Toc507275147)

[Conception 7](#_Toc507275148)

[Data structures 7](#_Toc507275149)

[Identities 7](#_Toc507275150)

[Users 7](#_Toc507275151)

[Global application flow 7](#_Toc507275152)

[Authentication and Menu 10](#_Toc507275153)

[The Operations Menu 11](#_Toc507275154)

[-->> User Operations: **Create** 11](#_Toc507275155)

[-->> User Operations: **Read** 12](#_Toc507275156)

[-->> User Operations: **Update** 12](#_Toc507275157)

[-->> User Operations: **Delete** 13](#_Toc507275158)

[-->> Identity Operations: **Create** 14](#_Toc507275159)

[-->> Identity Operations: **Read** 15](#_Toc507275160)

[-->> Identity Operations: **Update** 16](#_Toc507275161)

[-->> Identity Operations: **Delete** 17](#_Toc507275162)

[-->> File: Exit 18](#_Toc507275163)

[Global schema and major features schema 18](#_Toc507275164)

[Configuration instructions 20](#_Toc507275165)

[Prerequisites 20](#_Toc507275166)

[Schema creation 23](#_Toc507275167)

[Database Configuration 23](#_Toc507275168)

[Table 1: Various User Groups and their related operations they can perform 4](#_Toc507275091)

[Figure 1: The structure of the database showing the various schema and tables 6](#_Toc507275101)

[Figure 2: Sample SQL codes that was used in creating the tables 6](#_Toc507275102)

[Figure 3: The original application run flow of IACB 8](#_Toc507275103)

[Figure 4: The new program flow after analyzing the program requirements 9](#_Toc507275104)

[Figure 5: The login window which allows access to the IACB application 10](#_Toc507275105)

[Figure 6: The main IACB application window 10](#_Toc507275106)

[Figure 7: The IACB application menu 10](#_Toc507275107)

[Figure 8: The User Management window 11](#_Toc507275108)

[Figure 9: A window informing the user about an existing username 11](#_Toc507275109)

[Figure 10: Message window informing a user of a successful operation 12](#_Toc507275110)

[Figure 11: A search/read results of a particular Username 12](#_Toc507275111)

[Figure 12: List of user records with a particular record updated 12](#_Toc507275112)

[Figure 13: A window showing an update operation and message of success 13](#_Toc507275113)

[Figure 14: List of user records after update 13](#_Toc507275114)

[Figure 15: A window showing a user record which is about to be deleted based on the username 14](#_Toc507275115)

[Figure 16: A dialog asking if a delete operation is truly to be performed 14](#_Toc507275116)

[Figure 17:Main Identity Management window 14](#_Toc507275117)

[Figure 18: A message dialog informing a user of a duplicate unique id 15](#_Toc507275118)

[Figure 19: A message dialog showing an invalid e-mail information 15](#_Toc507275119)

[Figure 20: A message dialog information a successful create operation of an Identity 15](#_Toc507275120)

[Figure 21: A window showing a read operation based on the unique id of the identity and its subsequent results 16](#_Toc507275121)

[Figure 22: List of records before update of the Identity Test Test 16](#_Toc507275122)

[Figure 23: An update operation for the Identity Test Test and a message dialog indicating success 17](#_Toc507275123)

[Figure 24: The result of the update from Test Test to Thomas Broussard 17](#_Toc507275124)

[Figure 25: A delete operation based on the Identity unique id GH1002FR1 and its results after the operation 18](#_Toc507275125)

[Figure 26: A message is shown to inform the User of a successful delete option 18](#_Toc507275126)

[Figure 27: The IACB menu showing where the application can be closed 18](#_Toc507275127)

[Figure 28: A class diagram of classes in the services packages 19](#_Toc507275128)

[Figure 29: A class diagram in the exception package 19](#_Toc507275129)

[Figure 30: A class diagram of classes in the launcher package 19](#_Toc507275130)

[Figure 31: A class diagram of classes in the model package 20](#_Toc507275131)

[Figure 32: A window showing how the connection to the Derby database 21](#_Toc507275132)

[Figure 33: A diagram showing the list of database drivers for Derby and others 22](#_Toc507275133)

[Figure 34: A window showing the database connection details 22](#_Toc507275134)

[Figure 35: A window showing the structure of the database to of the application 23](#_Toc507275135)

# Subject Description

The I AM CORE BISMARK (code named IACB) is a first semester project presented to the faculty in the UML and Java Fundamental course at EPITA Graduate School of Computer Science and Advance Technologies, Paris-France.

This project is the implementation of an Identity Management Software. The project was partially developed during the semester’s class and has been completed during the last week during the end of the semester.

The implementation of this project has been done by command line only, no UI is provided.

# Subject analysis

## Major features

The IACB application is basically implemented to perform four elementary operations that most software applications do implement which is acronym as CRUD for (Create, Read, Update and Delete functionalities.

The application consist of two main modules which are the User Management (which manages the users of the IACB application and then the Identity Management (which also manages the actual identity records that can also be kept in the system) by the help of a database created in Apache Derby.

The following features are found in using the IACB application;

* Authentication

There should be a user account in the application before one can it. Since this is crucial, the application is developed such that at the first run of the application, even before the login window shows, it will check if there is/are User records in the database. If it finds no record, then it automatically creates a default User record with the following details;

* Username: admin
* Password: @123456u
* Enable-Login: 1 (for YES and 0 for NO)
* User Group: System Default
* Data Management Operations:

The application supports the following four (4) CRUD operations in any of the two modules which are the User Management and Identity Management.

* + Create

The create operation will allow such operation as INSERT (inserting data) into the appropriate database tables by the call of the required module method.

* + Read

The Read operation in IACB will help users of the application to read/search for records in the application in several ways either reading all records or specific records and results are returned to users.

* + Update

The Update method also will help in updating a specific record bases on the criteria that would be set in the SQL (server query language) UPDATE function.

* + Delete

The Delete operation is also allowed in the IACB application and will help to remove or a delete a record based a specified criteria set in the SQL statement.

As a security measure or practice in the IACB application, it is not all Users of the application that can perform all these CRUD operations that are list above. The following table shows the list of User Groups and the type of CRUD operations that each can perform.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| User Group | User Management Module | | | | Identity Management Module | | | |
| **Create** | **Read** | **Update** | **Delete** | **Create** | **Read** | **Update** | **Delete** |
| System Default | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Administrator | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Power User | 🗶 | 🗶 | 🗶 | 🗶 | ✓ | ✓ | ✓ | 🗶 |
| Reader | 🗶 | 🗶 | 🗶 |  | 🗶 | 🗶 | 🗶 |  |

Table : Various User Groups and their related operations they can perform

## Application Feasibility

The IACB application as already mentioned above, emanated from class learning with our teacher and most of the things done in class are same to be implemented in the application. It is feasible in the sense that the concepts of Inheritance, Polymorphism, Interfaces etc. and CRUD operations which were taught in class as same that application requires for implement.

These things learnt in class are not new from the IACB application even though the project requires an extension of the application from the console version to a graphical user interface (GUI) be it desktop or web user interface (UI).

This implementation of GUI will further help students to extend their learning from the class to finding solutions to their challenges outside the lecture halls.

## Data description

The handling of data on the two entities Users and Identities were done using the Apache Derby database with a Schema called FORTRESS and within this exists a database named IACBdb and within the table objects which are called tblUsers and tblIdentityBismark.

Users are stored into and retrieved from the tblUsers table and the Identities are as well stored and retrieved from the tblIdentityBismark table.

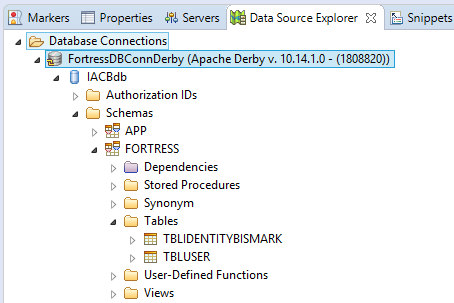


Figure : The structure of the database showing the various schema and tables

The figure below shows the structure of the database tables and their respective data types and the appropriate commands that were used in creating them for a quick reference.

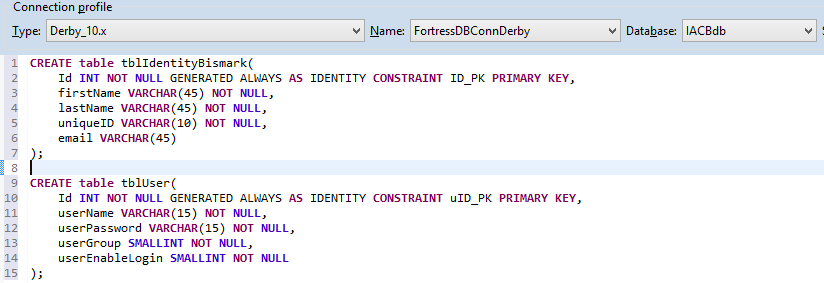


Figure : Sample SQL codes that was used in creating the tables

## Expected results

It is expected that in using the application IACB, a User first log-in to the application using his/her assigned credential before allowing access to perform any operation such operations of CRUD elaborated above.

Also, the required CRUD operations on Identities are all functional (though there may be bugs since the application may not be fully tested at completion).

Another expectation would be that only Users with a specific User Group performs such operation of CRUD assigned them.

## Scope of the application

The scope of the application was limited due to time constraints and other educational activities. But those factors aside, the application as it is now has been developed to manage both Users and Identities.

These management are done using only the GUI and the database access levels excluding other storage environments such as saving to Text and XML files.

As stated above, the application is limited to only manage Identities and not to manage the Users.

An improvement can be done later with more validation options and security features to enhance the IACB application.

# Conception

## Data structures

As stated at [Wikipedia](https://en.wikipedia.org/wiki/Data_structure), “in computer science, a data structure is a particular way of organizing and storing data in a computer so that it can be accessed and modified”.

In IACB, the data structure used was mainly to organize them first into classes with various fields and methods and then persisted to and from the Apache Derby database.

### Identities

This is the object in the application on whose properties or field are going to set and get and then saved into the application’s database based on the CRUD operations selected to be performed.

The structure of the identity can be referred to in Fig. 2

### Users

The entities are basically the persons who are going to use the application and their details required before using it.

Also you can refer to Fig. 2 to have a look at the properties which are core to them.

## Global application flow

Based on the understanding of the project questions and requirements, the analysis done brought about the following flow of how the application will be run.

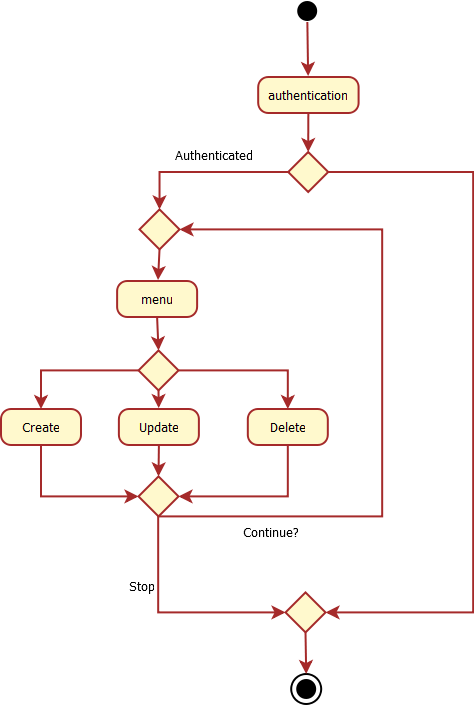


Figure : The original application run flow of IACB

The figure above shows the default application structure at the time of learning in class and then on starting up with the actual project and bases on my analysis, I came up with the below flow diagram of the IACB application.

This has been necessitated because, though student were to implement the practices of learning in class but we were supposed to extend the application in its functionalities hence the reason for this change and the results is the Figure 4 below.

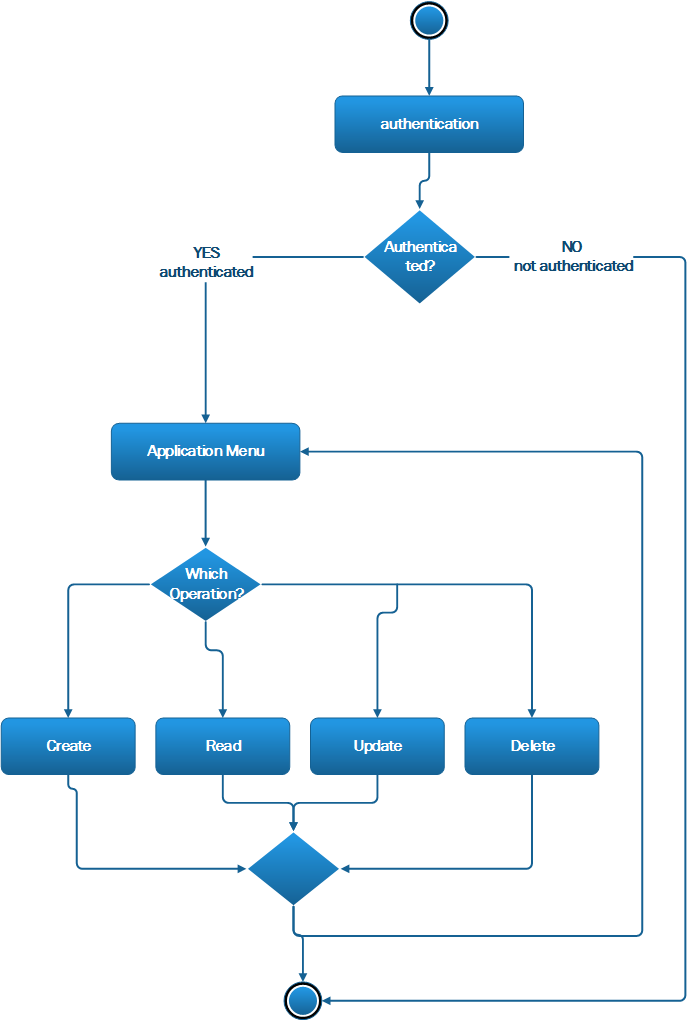


Figure : The new program flow after analyzing the program requirements

### Authentication and Menu

The application will be launched by entering correct Username and a matching Password before gaining access to the main application environment through the Login window.

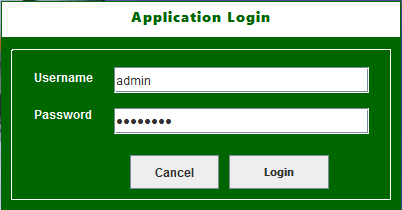


Figure : The login window which allows access to the IACB application

After a success authentication by entering a good login credentials, then the main window will open as show below.



Figure : The main IACB application window

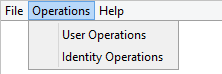


Figure : The IACB application menu

## The Operations Menu

### -->> User Operations: **Create**

Clicking on the Operations menu and selecting the User Operations will open up the Application User management window which has all the CRUD operation that has been described earlier above.

Once it is opened, the application will automatically load all Users record in the database into a control or components from the Java SWT library which includes called JTable. Other components in the library are JTextBox, JRadioButton, JComboBox, JFrame etc.

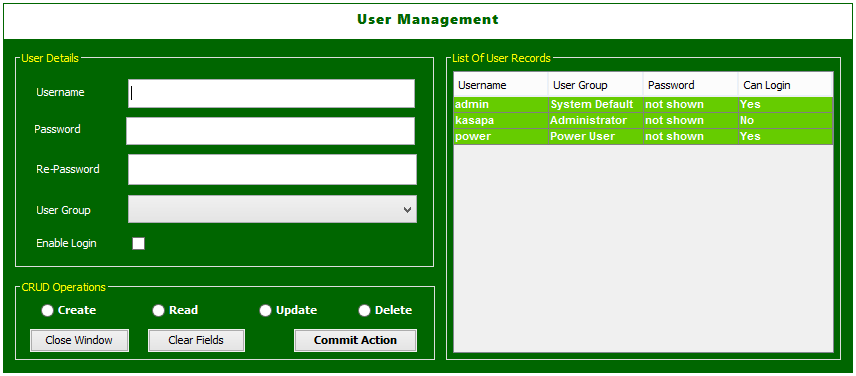


Figure : The User Management window

Validations are also done at this level such as;

* + Username existing already

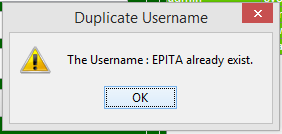


Figure : A window informing the user about an existing username

* + Un-matching password
  + Character length more than those specified in the database table definition, etc.

If no user group is selected, then a default user group of type Reader is assigned to the user account.

Based on the CRUD operation type that would be chosen, the appropriate action will be performed and indication of a success or failure would be shown to the user as depicting in the figure below.

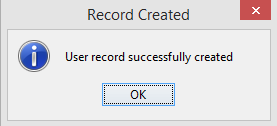


Figure : Message window informing a user of a successful operation

### -->> User Operations: **Read**

The Read operation is basically about searching or finding a record in the IACB application. All User groups within the application can do a search and it just simple. Jus select enter the search word and select the Read operation under CRUD Operations and then click on Commit Action to execute the read functionality.

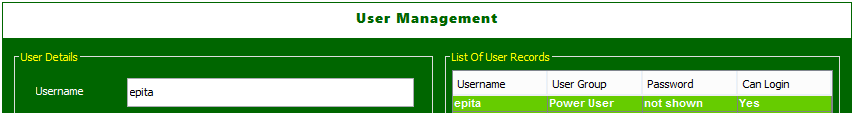


Figure : A search/read results of a particular Username

Because of time constraints and other requirement implementations, the Read operation under the User module was not fully implemented and would be considered a priority in the next version of the IACB life cycle.

### -->> User Operations: **Update**

Update is done on a User record when the correct details are entered especially the Username since the update is based on that field value matching what has been entered. What this means is that no duplicate Username is allowed in the User table.

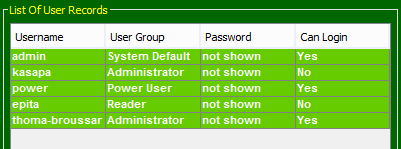


Figure : List of user records with a particular record updated

List of User records before update. Consider the User called epita in the list of users. It has a user group of Reader and login is disabled which is No.

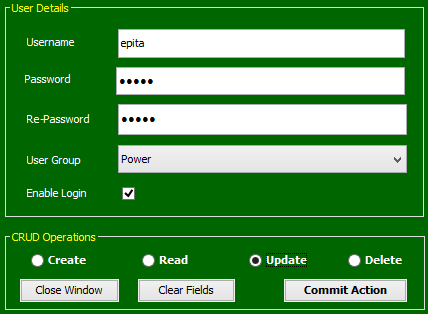
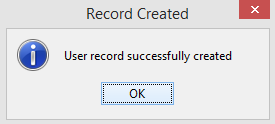


Figure : A window showing an update operation and message of success

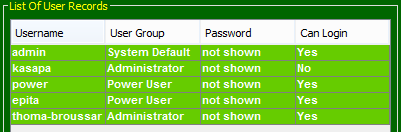


Figure : List of user records after update

Now look at the list of User records after the Update operation has been performed on the Username **epita**. It can be seen that the Username **epita** never got changed since it is the basis for the update.

### -->> User Operations: **Delete**

The Delete operation works similar to that of Create and Update operations. But here, it will not insert or modify and existing record but will instead remove or delete the record in question from the database

It can be seen from Figure [] that the record for the User **epita** exist but after performing a Delete operation using the Username field as the basis of the deletion, will remove it from the record as seen below.

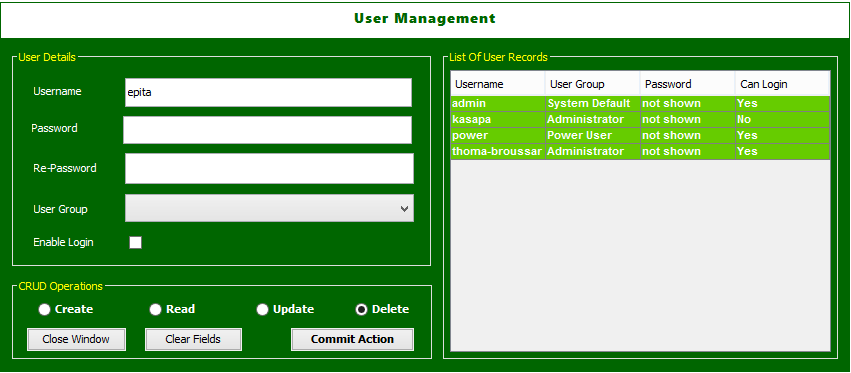


Figure : A window showing a user record which is about to be deleted based on the username

Performing a Delete operation is not straightforward as IACB assumes your deletion process may be an accident or unintentional, therefore it prompts the User if really he/she wants to execute such an operation. If the response is YES then deletion is done, else the operation is cancelled.

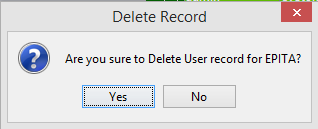


Figure : A dialog asking if a delete operation is truly to be performed

### -->> Identity Operations: **Create**

Clicking on the Operations menu and selecting the Identity Operations will open up the Identity Management window which has all the CRUD operations.

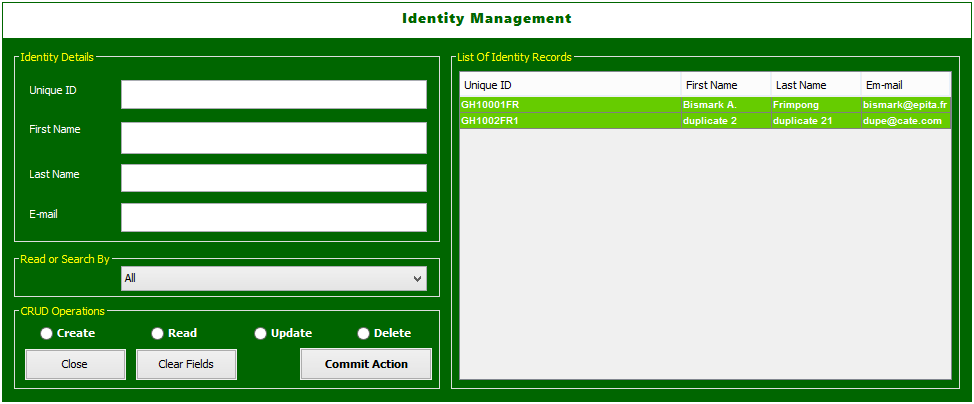


Figure :Main Identity Management window

Validations are also done at this level such as;

* + Unique ID existing already

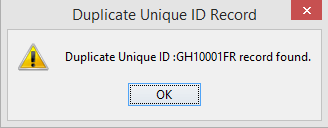


Figure : A message dialog informing a user of a duplicate unique id

* + Empty required fields
  + Character length more than those specified in the database table definition, etc.
  + E-mail address validations

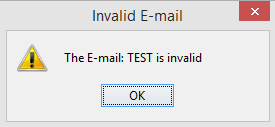


Figure : A message dialog showing an invalid e-mail information

Based on the CRUD operation type that would be chosen, the appropriate action will be performed and indication of a success or failure would be shown to the user as depicting in the figure below.

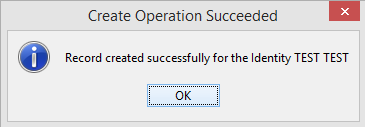


Figure : A message dialog information a successful create operation of an Identity

### -->> Identity Operations: **Read**

The Read operation in the Identity management is fully implement and to the extent that Users can search for records with different search options to choose from.

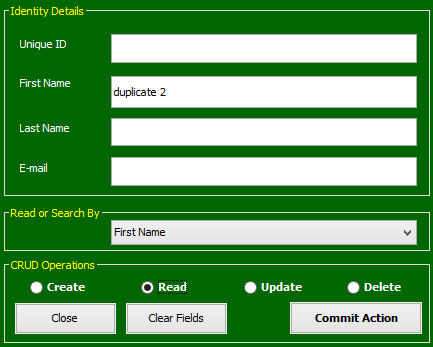
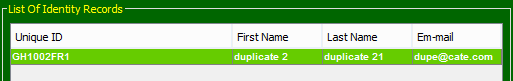


Figure : A window showing a read operation based on the unique id of the identity and its subsequent results

Users can search for records with options based on the follow;

* All – reads all records
* First Name – read/search based on the first name only
* Last Name – read/search based on the last name only
* E-mail – read/search based on the e-mail address only
* Unique ID – read/search based on the unique id only
* Fuzzy Search – read/search based on any of the above options including null values. It somehow the same as the **All** option

### -->> Identity Operations: **Update**

Update is done on an Identity record when the correct details are entered especially the Unique ID since the update is based on that field value matching what has been entered. What this means is that no duplicate Unique ID is allowed in the Identity table.

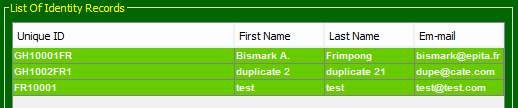


Figure : List of records before update of the Identity Test Test

List of Identity records before update. Consider the Identity called **Test Test** with unique id **FR1001** in the list of identities. It has a Test as both first and last names.

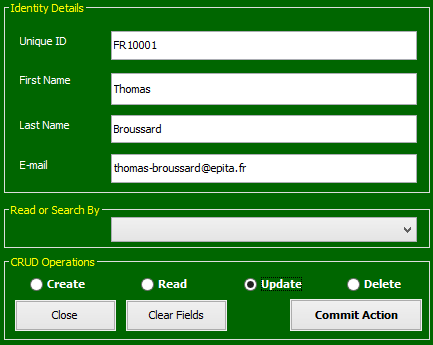
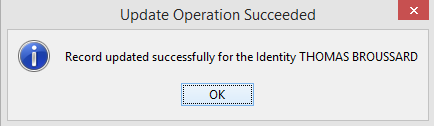


Figure : An update operation for the Identity Test Test and a message dialog indicating success

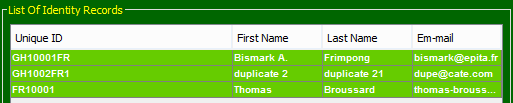


Figure : The result of the update from Test Test to Thomas Broussard

Now look at the list of Identity records after the Update operation has been performed on the Unique Id **FR10001**. It can be seen that the identity is now changed to **Thomas Broussard** but the unique id is still **FR10001**.

### -->> Identity Operations: **Delete**

The Delete operation works similar to that of Create and Update operations. But here, it will not insert or modify and existing record but will instead remove or delete the record in question from the database

It can be seen from Figure [] that the record for the identity **duplicate2 duplicate21** exist but after performing a Delete operation using the unique id field value of **GH1002FR1** as the basis of the deletion, will remove it from the records as seen below.

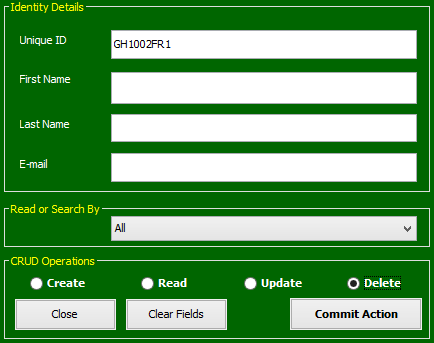


Figure : A delete operation based on the Identity unique id GH1002FR1 and its results after the operation

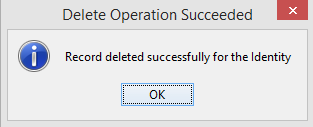


Figure : A message is shown to inform the User of a successful delete option

### -->> File: Exit

Clicking on the File menu from the IACB menu bar and selecting the Exit option will quit or exit the application.

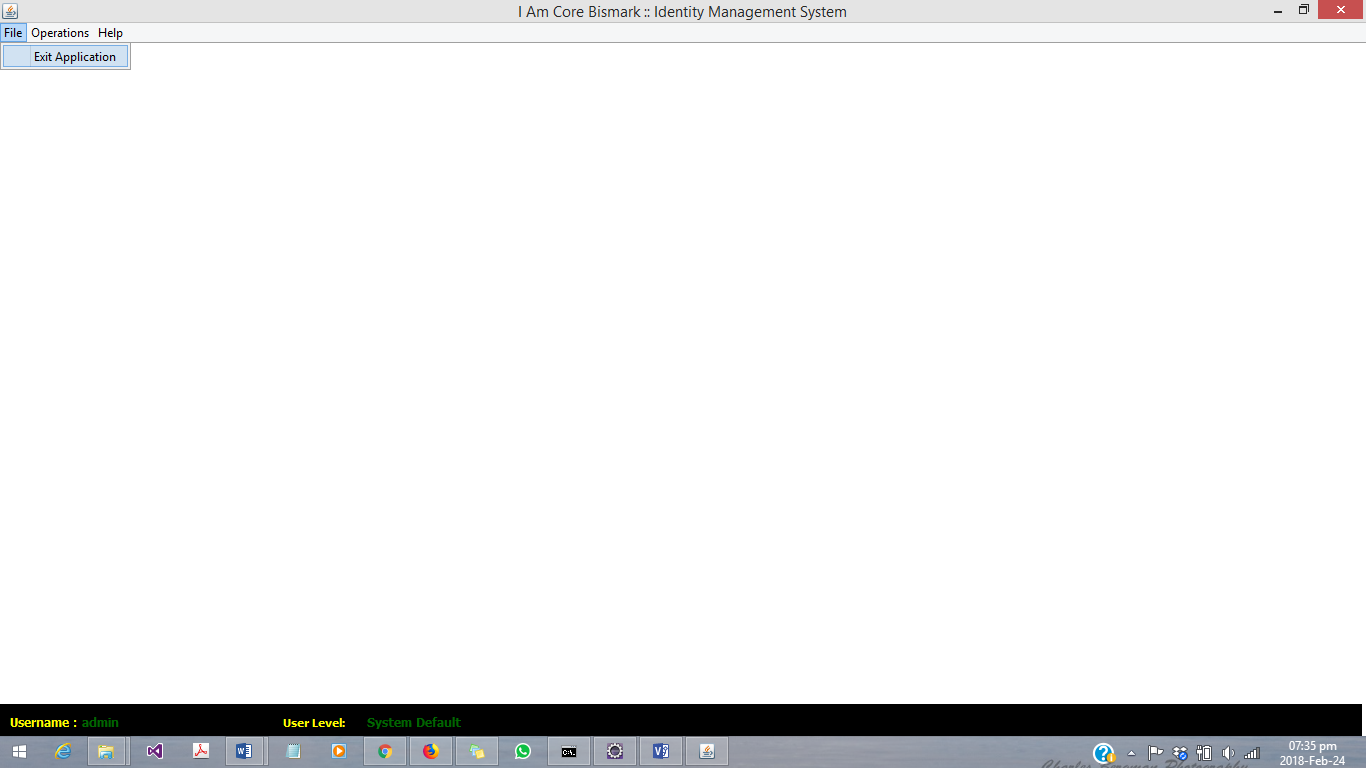


Figure : The IACB menu showing where the application can be closed

## Global schema and major features schema

The class diagram were generated by ObjectAid UML which is a library installed through the ecplise IDE.

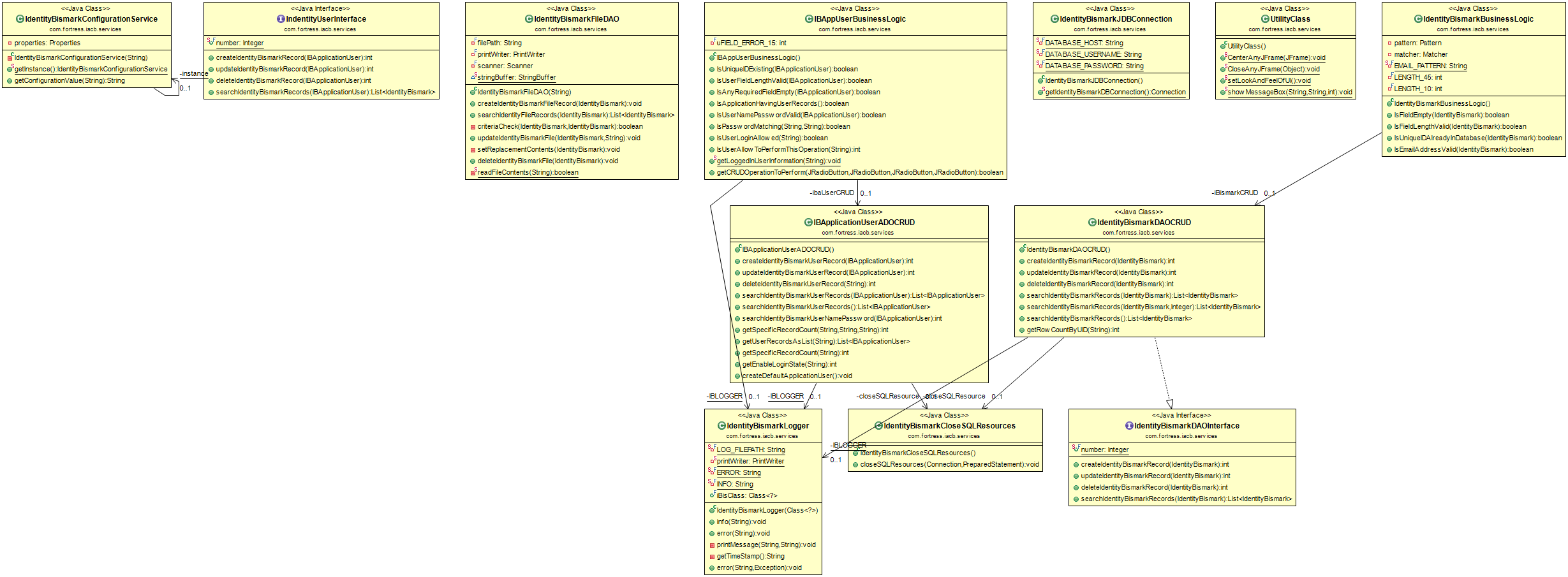


Figure : A class diagram of classes in the services packages

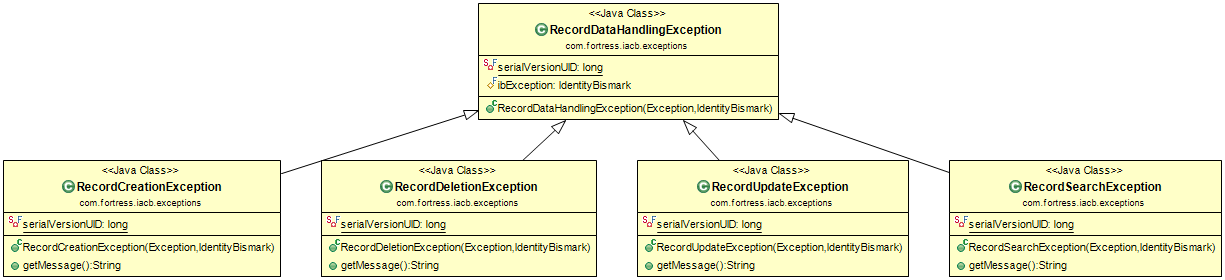


Figure : A class diagram in the exception package

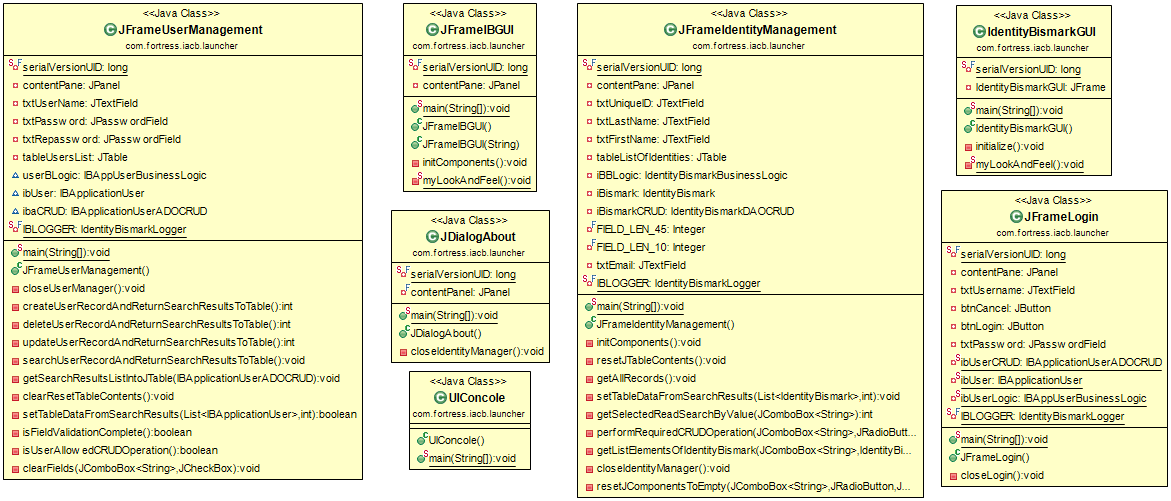


Figure : A class diagram of classes in the launcher package

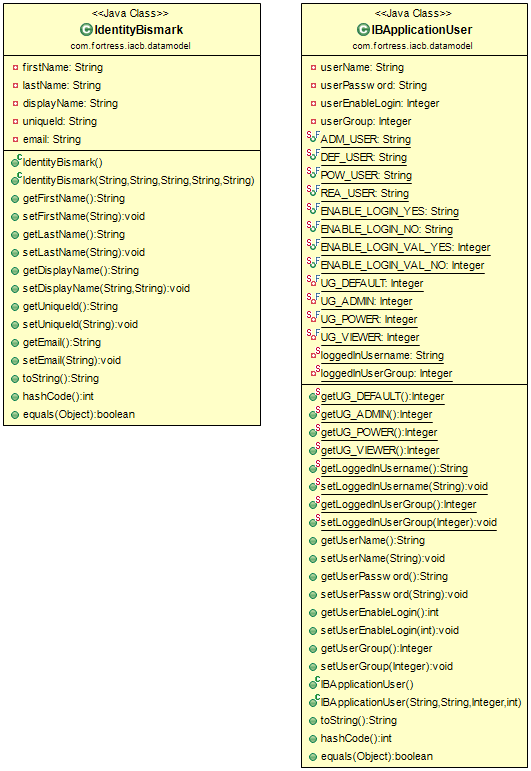


Figure : A class diagram of classes in the model package

# Configuration instructions

## Prerequisites

* Eclipse Oxygen IDE ( JEE version)
* Java Development Kit (JD)
* Derby library

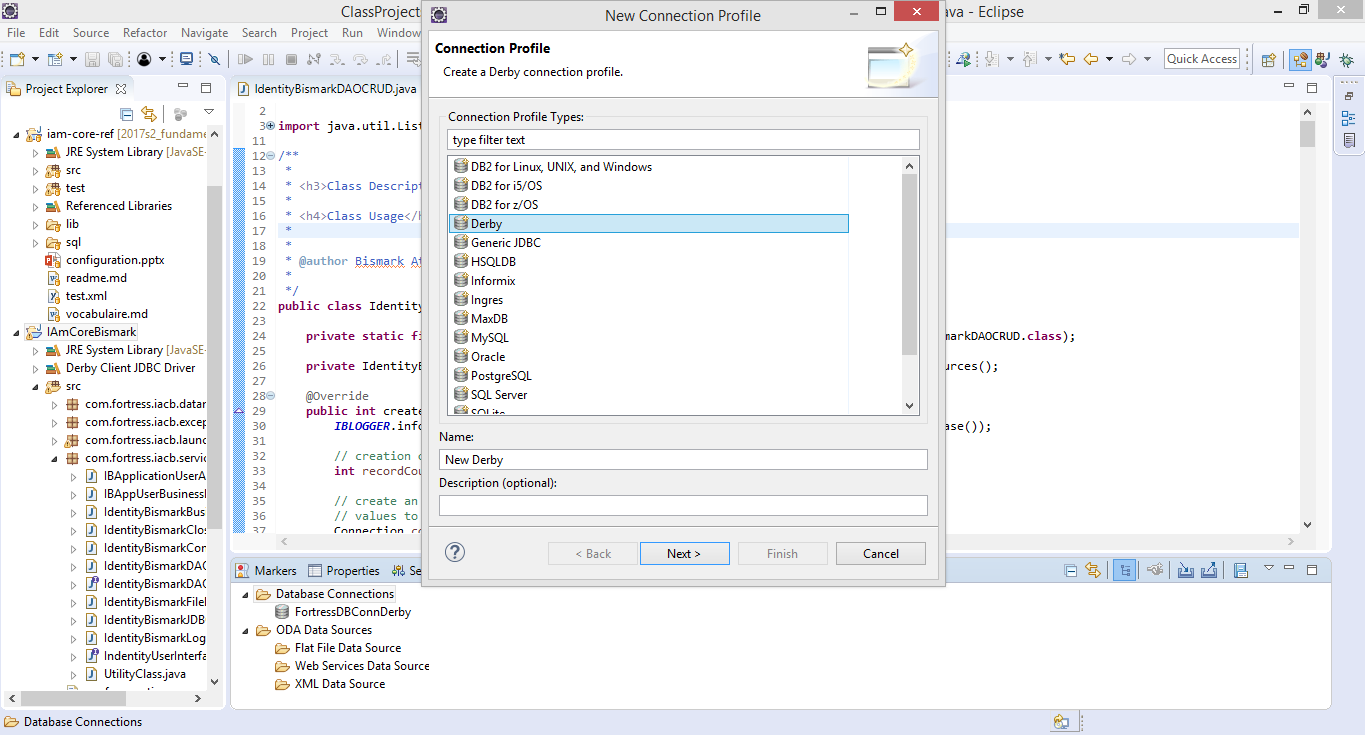


Figure : A window showing how the connection to the Derby database

1. Right under the Data Source Explorer, right-click on New
2. In the contextual window select Derby,
3. Enter the name for the connection and then Click Next.
4. In the new window click on the “New Driver definition button” 
5. Select any Definition on the Name/Type tab and then move to the JAR List tab.
6. On the JAR List tab click on Clear all button, so no files are listed. Then click on the Add and navigate to the lib folder inside your Derby database. Once there select the derbyclient.jar file.
7. Click Open, then the “New Derby Connection Profile” window should be updated. Please take note of the URL and change the user and password if desired. Remember those because we are going to need them later on.
8. Make sure the Derby Database is running and click on Test Connection button, it should succeed.
9. Click on Finish button.

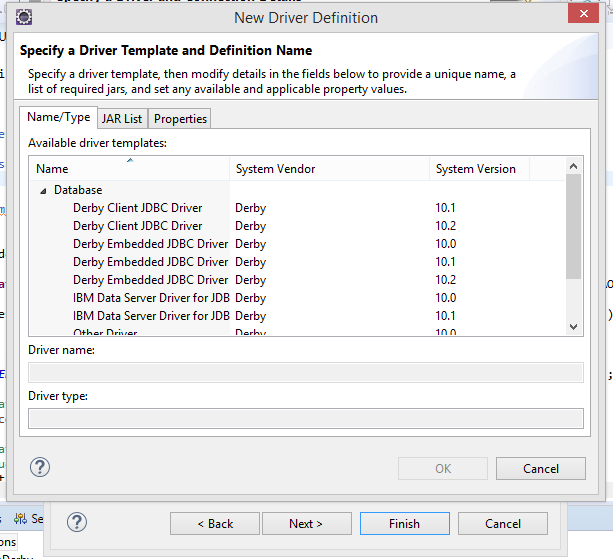


Figure : A diagram showing the list of database drivers for Derby and others

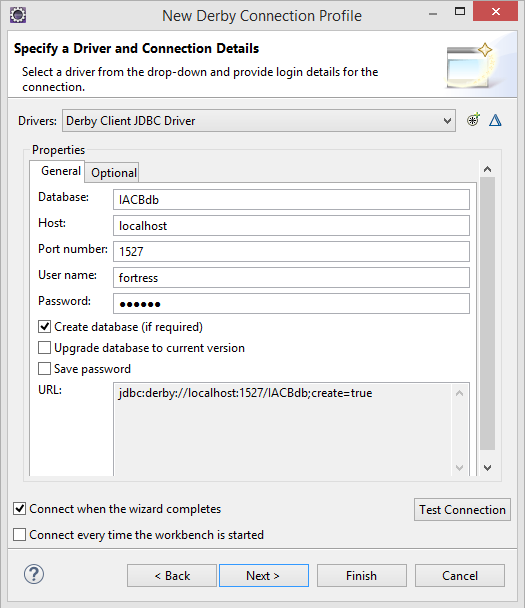


Figure : A window showing the database connection details

## Schema creation

1. Once the above process is done, within the Data Source Explorer window you should be able to expand all the way to Schemas:
2. Right click over the New Derby and select: Open SQL Scrapbook:
3. Once the new window opens, in the Project Explorer navigate: FortressDBConnDerby/FORTRESS/IACBdb/IdentityBismark table creation and open the file. Copy the file contents to the Scrapbook window and execute.
4. Right click anywhere on the Scrapbook window and select “Execute All”.
5. Execution should be successful:
6. Go back to the Data Source Explorer window and right click on the Schemas folder, select Refresh. You should now see and Shema with a name that matches your user name, inside it the Identities table should have been created:
7. Once all of this is done close the Scrapbook and the Identity table creation sq. windows.

## Database Configuration

1. Open the text file named conf.properties and change the details to suit your configuration
2. Modify the values with those according to your URL and user name and password:
3. Save the file.
4. To verify everything is ok, run the Test file.

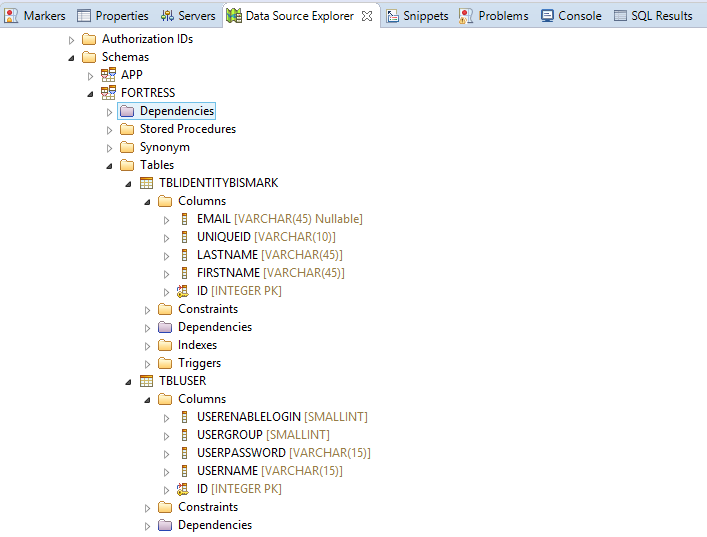


Figure : A window showing the structure of the database to of the application