

# IAM APPLICATION: THE IAM CORE PROJECT

## TEAM MEMBERS:

Stefano Acosta, Alvaro Bilbao.

## COURSE:

Java fundamentals.

**CONTENIDO**

SUBJECT DESCRIPTION .....	3
Subject analysis .....	3
Major features.....	3
APPLICATION FEASIBILITY.....	3
DATA DESCRIPTION .....	4
EXPECTED RESULTS .....	4
SCOPE OF THE APPLICATION (LIMITS, EVOLUTIONS) .....	4
CONCEPTION .....	5
DATA STRUCTURES.....	5
GLOBAL APPLICATION FLOW.....	5
GLOBAL SCHEMA AND MAJOR FEATURES SCHEMA.....	6
GUI (GRAPHICAL USER INTERFACE) OPERATIONS DESCRIPTION .....	7
USER AUTHENTICATION MENU.....	7
NEW USER REGISTRATION (WITH LINKED IDENTITY CREATION) .....	9
NEW IDENTITY CREATION.....	10
IDENTITIES MANAGEMENT (SEARCH, DELETE, UPDATE) .....	11
CONFIGURATION INSTRUCTIONS .....	13
BIBLIOGRAPHY .....	14

## SUBJECT DESCRIPTION

The general objective of this project is to deliver an Identity Management software, capable of providing an end user the possibility to manage identities in an information system (create, persist and manage them) through a friendly Graphical User Interface (GUI). The project is based on: Java as the coding language, DerbyDB as the database, GIT as a version control tool, and Eclipse Java EE IDE as development tool.

The specific objectives are:

Access, create and modify user information.

Persist users data in a database.

Be robust, capable of good performance.

Propose a simple but efficient user interface.

## SUBJECT ANALYSIS

### MAJOR FEATURES

- User authentication
  - Provide means to create a user with at least a unique username and a password.
  - Attach an Identity to the user, since the user itself is an Identity
  - Allow any user who has a valid account to login into the Identities management system.
- Identities Management
  - Provide services to create an Identity and save it in the database
  - Provide the user with the option to manage existing Identities
    - Update
    - Delete
    - Search by fields
- GUI
  - Provide a friendly and intuitive graphical interface, that allows the user to authenticate and use the software.

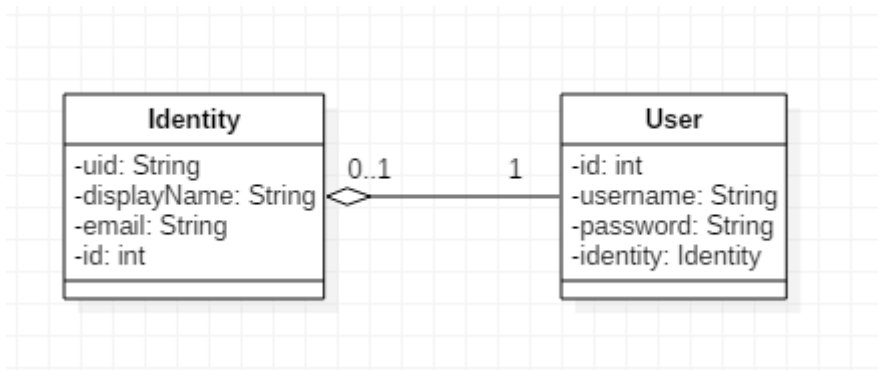
### APPLICATION FEASIBILITY

The application is well defined, we know beforehand all the data types that can be used during the creation of both the identities and the users. The database structure, and the interaction between objects is defined inside the application. We also know the identities and user attributes.

The only unknown factor for the application is the actual data that will be entered by the users of the application, however this is controlled with the data types.

## DATA DESCRIPTION

This application handle 2 things Users and Identities, we have made the following model to conceive them.



As we can see an identity has 4 attributes, the uid, the displayName and the email, which are values known to the user, and we also have an id field which contains the id of the identity in the database.

On the other hand, we have the User, he has an id, which is its identifier in the database, it has a unique username and a password, which cannot be null. And it also has one identity attached to it, this identity, can also be null and if it is erased, it will be set to null and the user will remain there (Aggregation relation).

## EXPECTED RESULTS

We expect that the user is able to perform the following operations over Identities:

1. Create
2. Read
3. Update
4. Delete

And he can also perform these operations on users:

1. Create
2. Login

And all these changes should be preserved in a Database.

## SCOPE OF THE APPLICATION (LIMITS, EVOLUTIONS)

The application is a desktop java application, it is limited to perform only one action at any given moment per user, this means that a single user cannot perform more than one identities creation at a given time. If the user wants to create multiple identities, he must create them one by one, the same happens for the update and delete capabilities.

The user authentication is weak, it can be improved by using hash over the password field, instead of storing it in plain text.

All the fields are limited to a max of 255 characters long.

The identities notion can be extended to store any attribute that a user wants, and different identities could have different attributes.

## CONCEPTION

### DATA STRUCTURES

For handling the users and identities, we have defined our own classes that represent them as shown in the data description schema.

To save the data, we are using DAO (Data access object), we have both a DAO for users, and another DAO for Identities.

A DAO is a data type that provides an interface to persist data into a database, without exposing the database itself, it provides methods for:

- Create (save something in the database)
- Read (search for the data that exists in the database)
- Update (modify something that already existed in the database)
- Delete (remove something from the database)

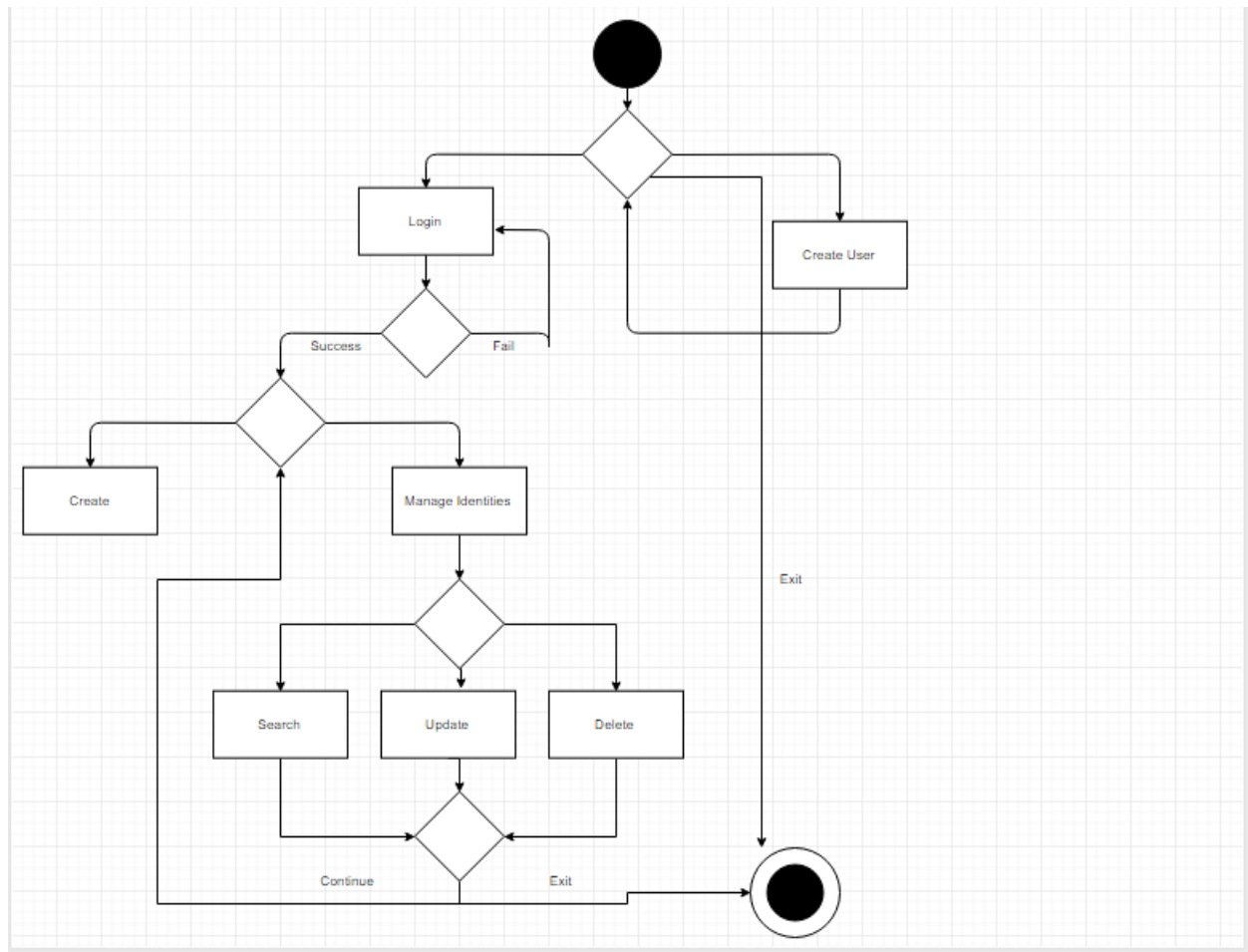
The Identities DAO provide functions to perform the 4 operations, while the users DAO only exposes the Create, read and delete functions, even though that the delete function has been added only for testing purposes.

This DAO methods make use of other data structures like Lists.

To communicate with the database, we use a JDBC object, which relies on the derbyClient driver provided by oracle.

### GLOBAL APPLICATION FLOW

The data flow is defined by the following scheme.



The user starts the application and he can create users or login to the system.

If the login is rejected, he can retry.

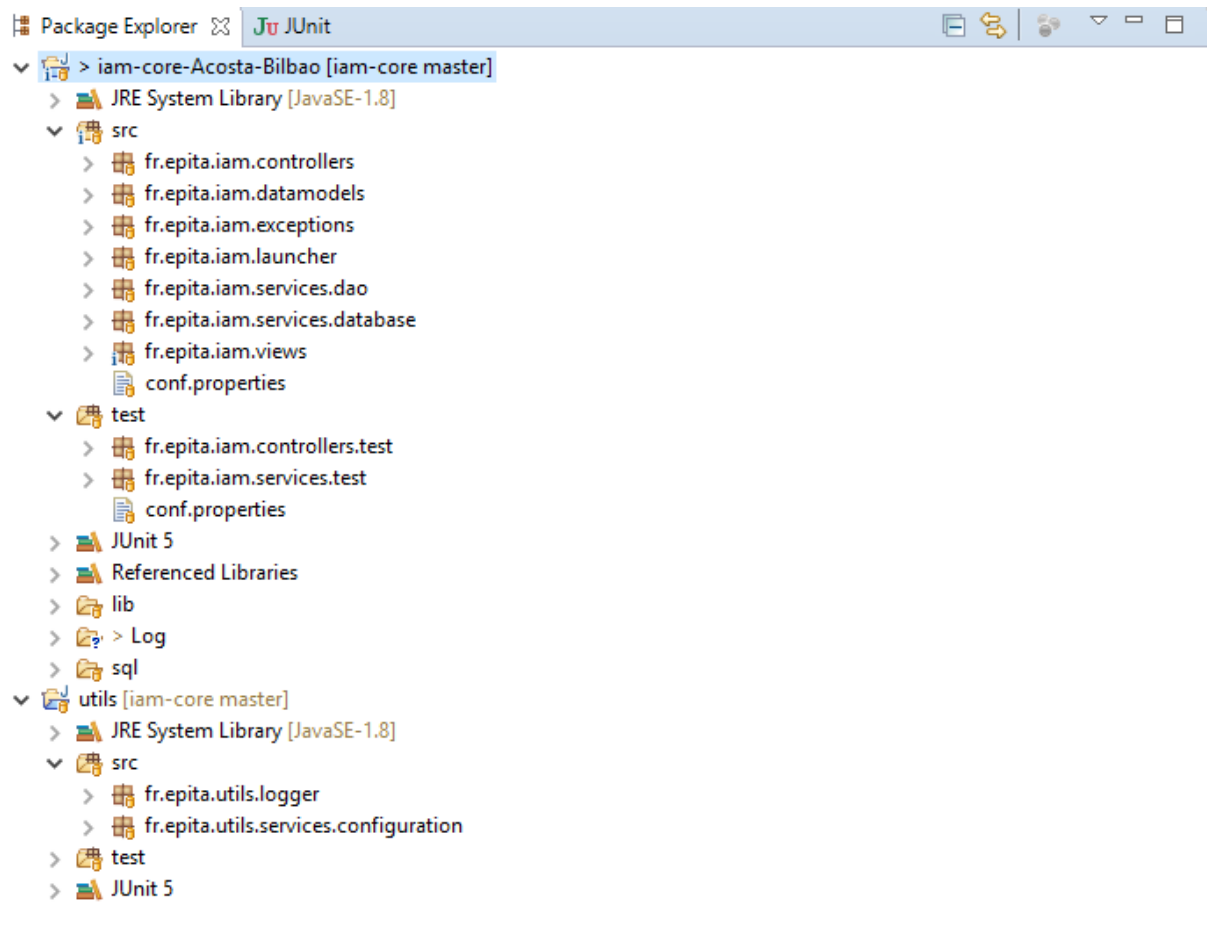
Once he has logged in he will be able to create and manage the identities. At any given moment he can close the application.

After any operation that performs a change in the data that is being stores, a Database connection is set to save these changes and preserve them.

#### GLOBAL SCHEMA AND MAJOR FEATURES SCHEMA

The application consists of 2 projects, one which is actually the application itself, and 1 small project which has some classes that can be used for the application to perform easy repetitive actions, this utilities project can be seen as a library to handle the project.

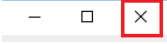
The schema for the project organization is the following.



As it is a GUI application, we decided to use an MVC kind of organization with datamodels, that represent the Database structure where both Identities and Users are saved. We have views which represent the GUI. We also have a controllers package which will handle the events related to their views.

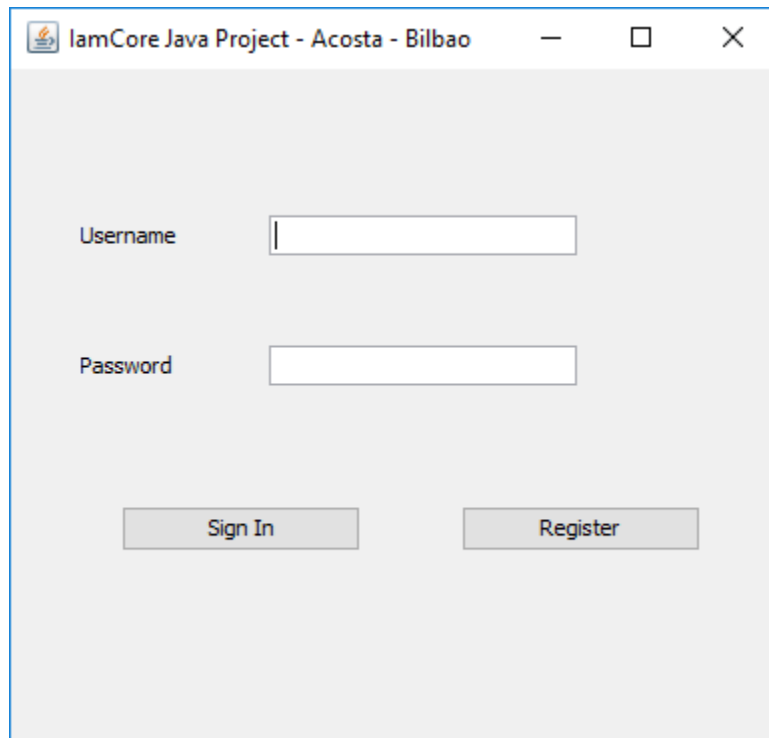
The other important part are the services, which will actually bring the logic to preserve and modify the identities.

## GUI (GRAPHICAL USER INTERFACE) OPERATIONS DESCRIPTION

\*\* General observation: Every window has the  "exit" option available for closing the software.

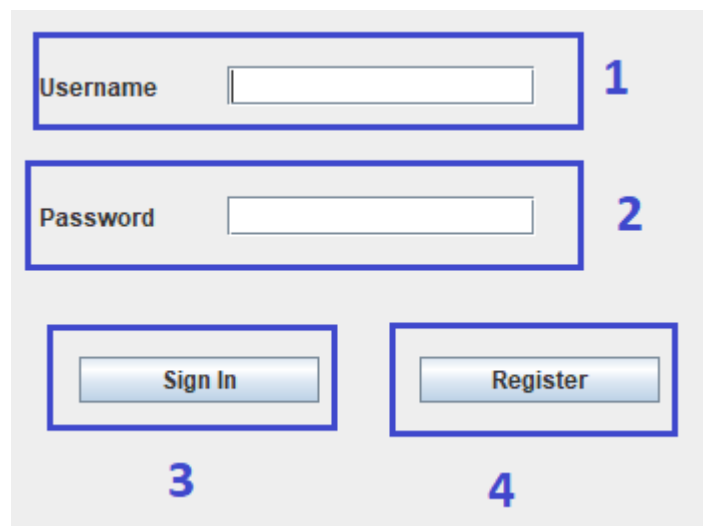
## USER AUTHENTICATION MENU

This is the main frame view, here the user will be able to authenticate (if registered in the database) or to register as a new user:



The screenshot shows a Java Swing window titled "IamCore Java Project - Acosta - Bilbao". Inside the window, there is a login form. It consists of two text input fields: one labeled "Username" and one labeled "Password". Below these fields are two buttons: "Sign In" and "Register". The window has a standard title bar with minimize, maximize, and close buttons.

The description goes as follows:



The diagram shows the same login form as the screenshot, but with blue rectangular boxes and numbers indicating specific steps or elements:

- 1**: A blue box around the Username field.
- 2**: A blue box around the Password field.
- 3**: A blue box around the Sign In button.
- 4**: A blue box around the Register button.

To Sign In, the user has to follow these steps:

- Fill in the Username field (1).
- Create a password in Password Field (2).
- Click on the button Sign In (3), to authenticate the entry, then the "New Identity Creation" window will be opened.

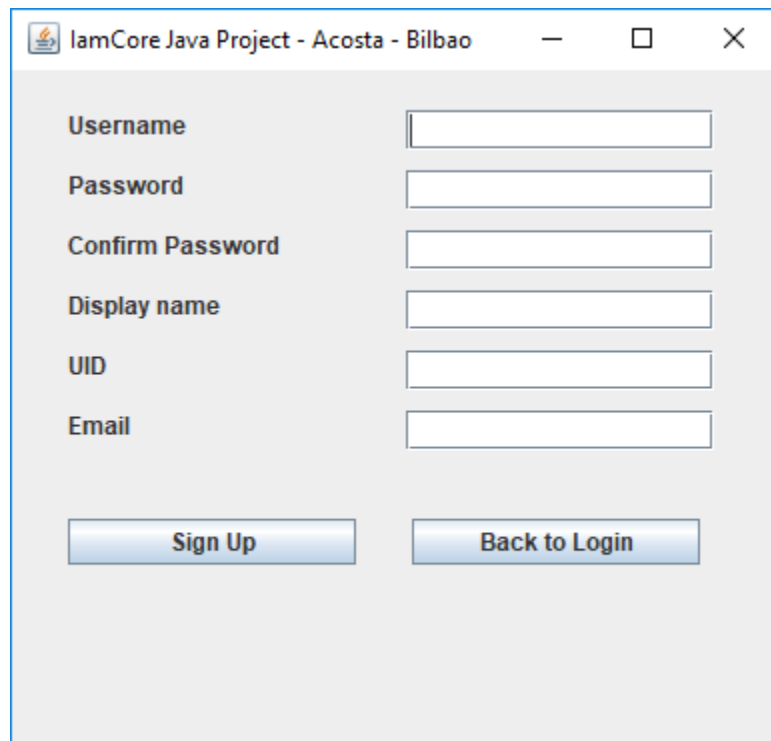
To register:

- Click on the Register button (4), a new window will be opened (instructions continue in "NEW USER REGISTRATION" description).



### NEW USER REGISTRATION (WITH LINKED IDENTITY CREATION)

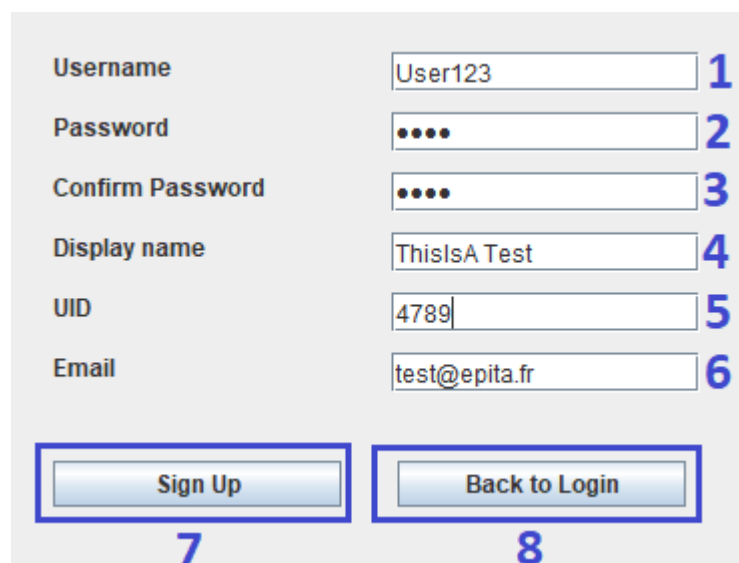
This is the user registration view, here the user will be able to register as a new user in the database, considerations are taken such as password consistency entry, the user may or may not fill the additional identity fields:



The screenshot shows a web application window titled "IamCore Java Project - Acosta - Bilbao". Inside the window is a registration form with the following fields and buttons:

- Username:
- Password:
- Confirm Password:
- Display name:
- UID:
- Email:
- Buttons: "Sign Up" and "Back to Login"

The description goes as follows:



The annotated form shows the following values and annotations:

- Username: User123 (1)
- Password: .... (2)
- Confirm Password: .... (3)
- Display name: ThisIsA Test (4)
- UID: 4789 (5)
- Email: test@epita.fr (6)
- Buttons: "Sign Up" (7) and "Back to Login" (8)

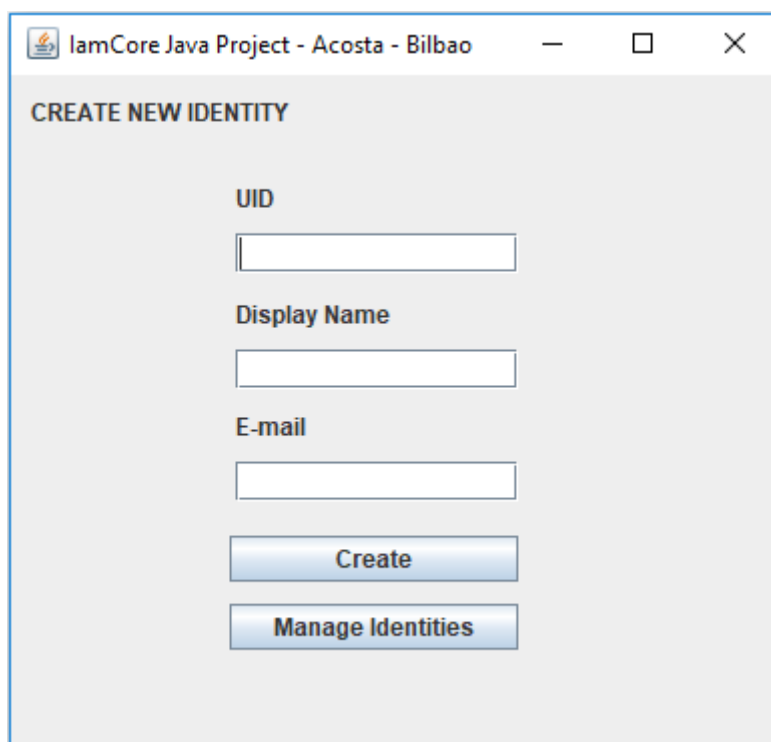
To create the desired new user account and Sign Up, the user has to:

- Fill the fields: Username (1), password (2), confirm password (3) with the same content as in (2).

- Fill the remaining fields (4)(5)(6) as considered necessary (it would be optimal to fill all the fields to create consistency).
- Click on the button Sign Up (7) to register and create the user, then the user will be redirected to the "user authentication menu (main menu)" or click on button Back to Login (8) to cancel the operation and return to the "main menu".

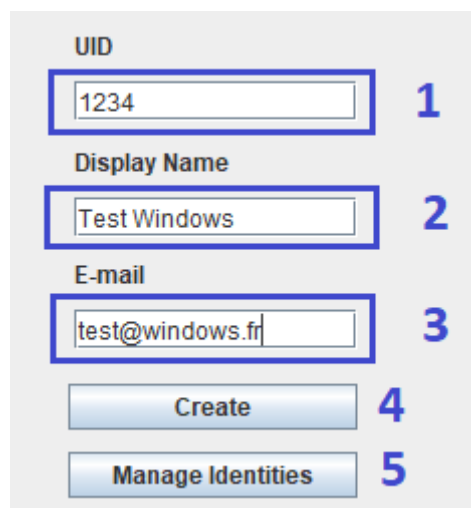
## NEW IDENTITY CREATION

This is the identity creation view, here the user will be able to create as many identities (not user linked) as needed. Each identity has a unique ID index in the database so even if all the fields are the same for two or more identities, there would not be any conflict:



The screenshot shows a window titled "IamCore Java Project - Acosta - Bilbao" with a close button. Inside the window, the title "CREATE NEW IDENTITY" is displayed. Below the title, there are three input fields: "UID", "Display Name", and "E-mail". Each field has a corresponding text input box. Below the input fields, there are two buttons: "Create" and "Manage Identities".

The description goes as follows:

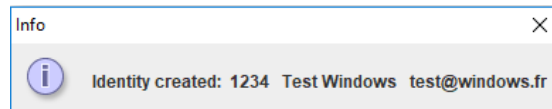


The annotated form shows the following elements with numbered blue boxes and numbers to the right:

- 1. UID input field (containing "1234")
- 2. Display Name input field (containing "Test Windows")
- 3. E-mail input field (containing "test@windows.fr")
- 4. Create button
- 5. Manage Identities button

To create a new identity, the user has to:

- Fill the fields (as needed): UID (1), Display Name (2), E-mail (3).
- Click on the button Create (4) to persist identity in the database, then repeat the process as needed for creating other identities, each time an identity is created a pop-up will appear confirming the creation.



To manage the created identities, the user has to:

- Click on Manage Identities (5), the user will be redirected to "Identities Management" window (indications continue in the next section).

### IDENTITIES MANAGEMENT (SEARCH, DELETE, UPDATE)

This is the identity management view, which will allow the user to search, delete or update persisted identities from the database:

A screenshot of a Java application window titled 'IamCore Java Project - Acosta - Bilbao'. The window contains a form for managing identities. On the left, there are three input fields labeled 'UID', 'Display Name', and 'E-mail' with the instruction 'For searching an identity, please fill the following fields:'. Below these fields is a 'Search' button. In the center is a large empty rectangular box. Below this box is the instruction 'To delete, search an identity, then click on it on the list, press' followed by a 'Delete' button. On the right, there are three input fields labeled 'UID', 'Display Name', and 'E-mail' with the instruction 'To update, search an identity, then click on it on the list.'. Below these fields is an 'Update' button. At the bottom center, there is a 'Back to create' button.

The description goes as follows:

For searching an identity, please fill the following fields:

UID

Display Name

E-mail

Search

Delete

Back to create

To delete, search an identity, then click on it on the list, press

To search an identity, the user has to:

- Fill the fields (as needed): UID (1), Display Name (2), E-mail (3) or leave everything blank to retrieve all the existing identities.
- Click on the button Search (4) to look for identities that match the fields in the database, then repeat the process as needed for searching other identities, the identities matching the fields will appear as a list in (5).

2345	Test User	testUser@tur.com
2345	Test User	testUser@tur.com
1234	Test Windows	test@windows.fr
7777	Test Updated	s@up.com
2333	Test Updated	s@up.com
9999	Test User	testUser@tur.com
2345	Test User	testUser@tur.com
2345	Test User	testUser@tur.com
2345	Test User	testUser@tur.com
2345	Test User	testUser@tur.com
2345	Test User	testUser@tur.com

Search

Delete

Back to create

To delete, search an identity, then click on it on the list, press

To delete an identity, the user has to:

- Do any search to show identities to be managed (deleted in this case), select one identity from (5) by clicking on it as (6), then click on Delete (7) to delete the identity from the database.
- If back to create (8) is pressed, the user will return to Identity creation window.

The screenshot shows a web application for managing users. It features a table of users, a search and update section on the right, and action buttons at the bottom. Numbered annotations highlight specific elements:

- 5**: Points to the table of users.
- 6**: Points to a selected row in the table.
- 7**: Points to the 'Delete' button.
- 8**: Points to the 'Back to create' button.
- 9**: Points to the 'UID' label and input field.
- 10**: Points to the 'Display Name' label and input field.
- 11**: Points to the 'E-mail' label and input field.
- 12**: Points to the 'Update' button.

Table content:

2345	Test User	testUser@tur.com
2345	Test User	testUser@tur.com
1234	Test Windows	test@windows.fr
7777	Test Updated	s@up.com
2333	Test Updated	s@up.com
9999	Test User	testUser@tur.com
2345	Test User	testUser@tur.com
2345	Test User	testUser@tur.com
2345	Test User	testUser@tur.com
2345	Test User	testUser@tur.com
2345	Test User	testUser@tur.com

Update section:

To update, search an identity, then click on it on the list.

UID: 9999

Display Name: Test User

E-mail: testUser@tur.com

Action buttons: Delete, Back to create, Update.

To update an identity, the user has to:

- Do any search to show identities to be managed (updated in this case), select one identity from (5) by clicking on it as (6), then the fields for the Update section (9, 10, 11) will be filled automatically, modify fields as needed and click on Update (12) to update the identity in the database, the list will be refreshed automatically.

## CONFIGURATION INSTRUCTIONS

- The application needs a configuration file to be sent as a system parameter, this configuration file should contain the following data in the form key=value:
  - db.host=(url for the database, for instance)
  - db.user=(database user)
  - db.pwd=(database password for the user)
  - log.path=(file path where the log file will be stored)
- As the project needs to run over a database, you will have to install and run DerbyDB
  - <http://apache.mindstudios.com//db/derby/db-derby-10.14.1.0/db-derby-10.14.1.0-bin.zip> (Download link)
  - Run the create.sql that is provided with the project to create the DB and the tables
- To run the application, execute the following command on the directory where the jar file and configuration file are located.
  - java -Dconf=conf.properties -jar iamCore.jar

## BIBLIOGRAPHY

- Junit5 tutorial, extracted from the web:

<< <https://howtoprogram.xyz/java-technologies/junit-5-tutorial/> >>, on 13/02/2018.

- JavaDoc quick tutorial, extracted from the web:

<< [https://www.tutorialspoint.com/java/java\\_documentation.htm](https://www.tutorialspoint.com/java/java_documentation.htm) >>, on 16/02/2018.

- SQL commands for DerbyDB, explanation and examples, extracted from the web:

<< <https://db.apache.org/derby/docs/10.8/ref/> >>, on 14/02/2018

- Oracle, UI with Swing tutorial, extracted from the web:

<< <https://docs.oracle.com/javase/tutorial/uiswing/> >>, on 16/02/2018.