Identity Access Management Software

Project Documentation

Implemented By

Hoang Anh DOAN

Favio TEJADA

Version: 1.0

Table of Contents

1. Purpose

Nowadays, Java is one of the most popular programming languages which are used for developing software. This Identity Access Management application is implemented in order to discover how to program in Java as well as learning how to use basic and supportive features, advantages and tools of Java such as Developer Kit APIs, databases and Eclipse.

The result of this implementation covers some key knowledge of starting a Java project from scratch, applying OOP programming model, following some design patterns as well as exploiting the tool Eclipse.

2. IAM description

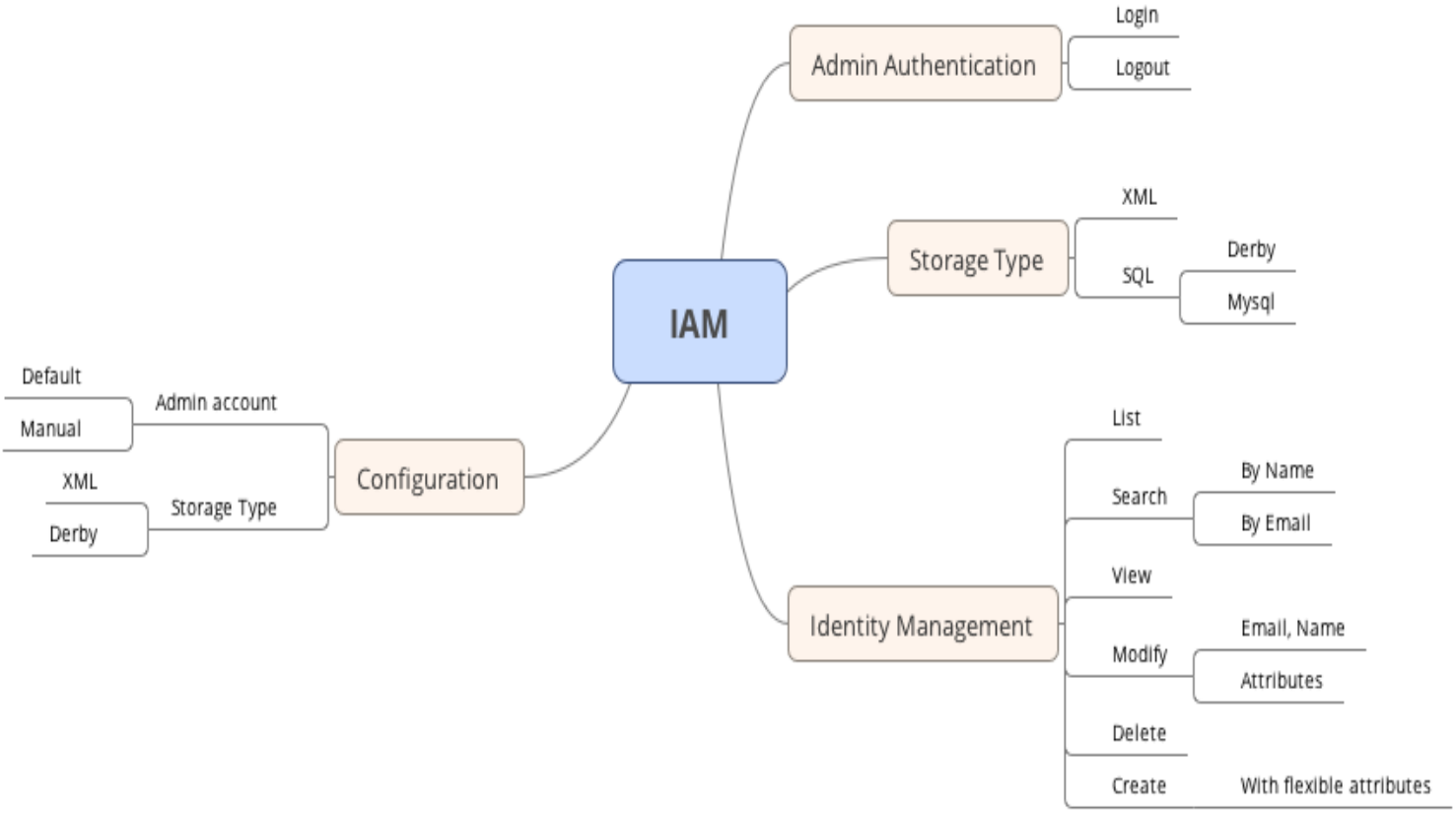
The IAM is a desktop application for managing identities. It allows administrator to list the identities, create new identity, modify and update identities.
The core application is designed to work with many types of User Interface like console, web and desktop application. In IAM, the management can be done in an easy way with a supportive and convenient UI.
It is also prepared to communicate with different types of storage implementation like XML and SQL Databases (Derby, Mysql,…) with easy configuration and installation.

3. IAM analysis

3.1. Major features:

- Admin authentication
- Identity Management (List, Search, Create, Modify, Delete)
- Storage type options
- Easy Configuration

The following diagram represents the list of functionalities provided by the IAM Software:



3.2. Application Feasibility

- Technological Feasibility: This application can be built on Java – a powerful programming language using popular data storages such as Files (XML) and Relational Database (like Derby, MySQL) with the help of Eclipse tool. The functionalities of this application is basic management, the concept of it is very popular and easily understandable.
- Economic Feasibility: Java and some database management applications are open-source so we can use them without any charges.
- Other areas like legal or organizational feasibility: The technology, resources and tools are open-source, so we are free to use legally.

3.3. IAM Objectives:

- A desktop application with five UI views: login, menu, identity list & search, identity detail & modify & delete and identity create views.

- The application can work well with both File (XML) and Database (Derby) storages.
- Allow the user to set the simple configuration.

3.4. Data description:

In this application, we have two types of objects to deal with, which are Identity and User.

Identity:

- Uid: is a string represents an unique identity
- Name: is a string represents the name (display name) of an identity.
- Email: is a string represents and email address of an identity.
- Attributes: is a set of key-value strings represents additional information of an identity (age, address, gender, …).

User:

- Uid: is a string represents a unique identity
- Name: is a string represents the name (display name) of an identity.
- Email: is a string represents and email address of an identity.
- Attributes: is a set of key-value strings represents additional information of an identity (age, address, gender, …).
- Password: is a secret string which combines with name to form a credential (for admin authentication).

3.5. Algorithm study

In IAM, the admin (user) can list all the identities or search for a specific identity. When there are a lot of identities, the question is how to get all the identities or search for an identity without making the admin wait for a long time or a powerful computer (CPU, RAM) is required. And when displaying identities, how we can sort the identities with some criteria effectively if we need to.

Hence, for list and search operations, we need an effective algorithms to achieve the expected performance.

3.6 IAM scope, limitation and evolution

3.6.1. Scope

- Desktop application with window forms.
- Identity management (list, search, create, delete, modify) with name, email and some additional attributes.
- Admin authentication (with pre-created admin account)
- Outside scope: User management.

3.6.2. Limitation:

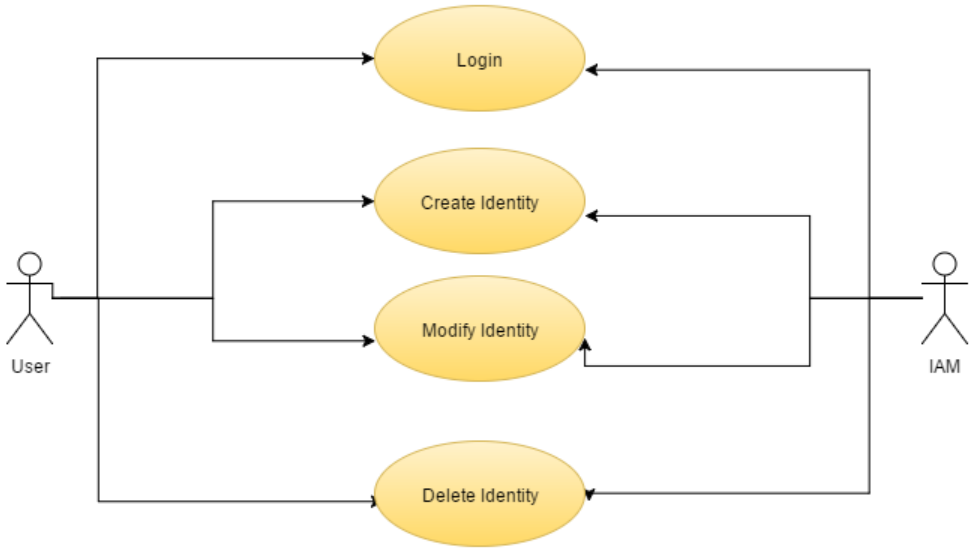
- Must be installed on a computer.
- Work as offline-application.
- Take time to change the UI.

3.6.3. Evolution:

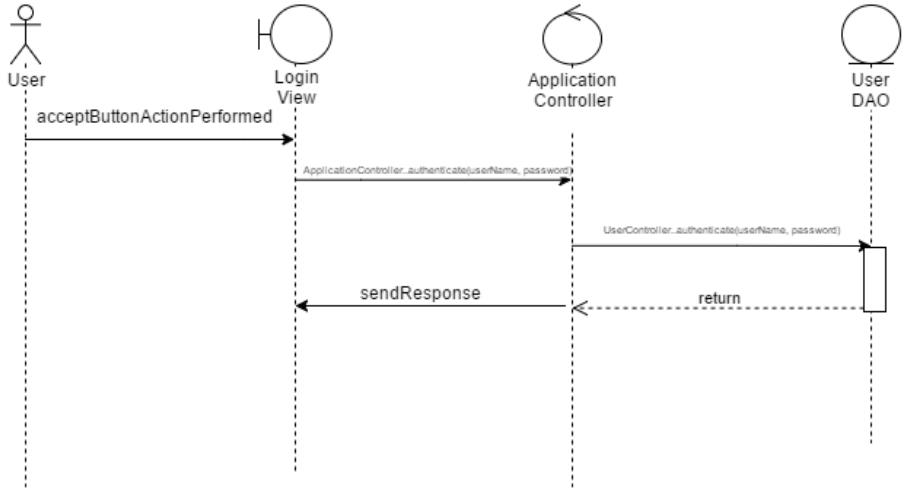
- More options for searching identity.
- Pagination when listing identities.
- User management.
- More graphical UIs.
- More flexible designs for expansion.
- Support more data storage types (SQL like Mysql, Postgres … and NoSQL like Cassandra, MongoDB).
- Enable installation option.

3.7 UML Design

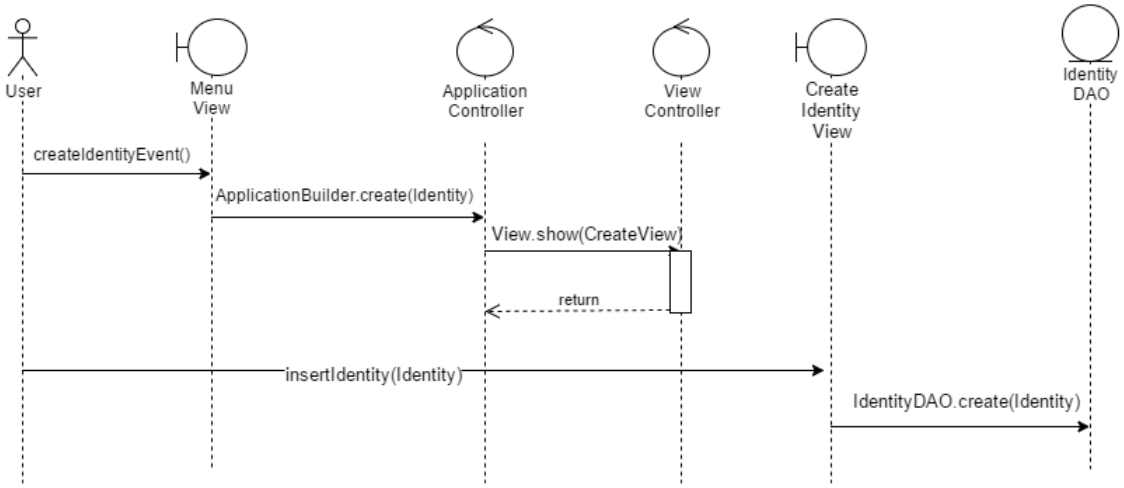
3.7.1 Uses case



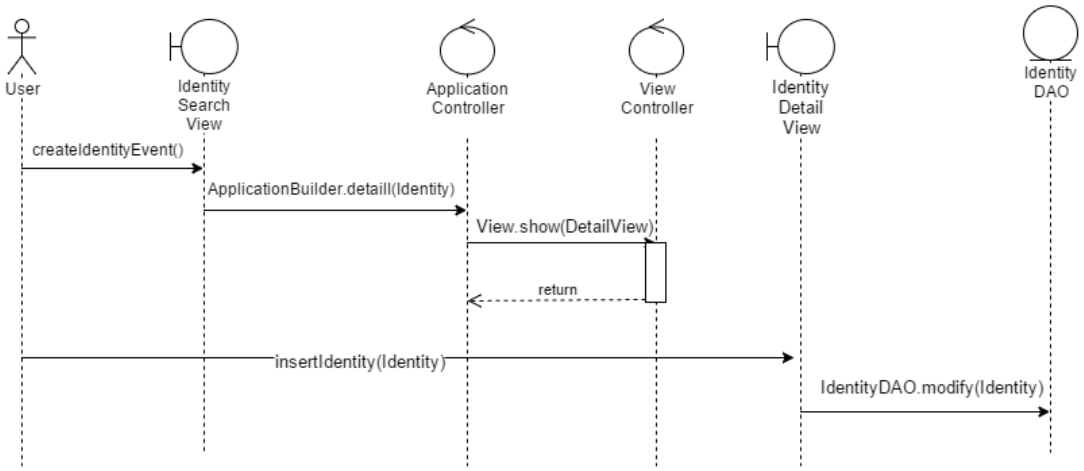
3.7.2 Sequence Diagrams



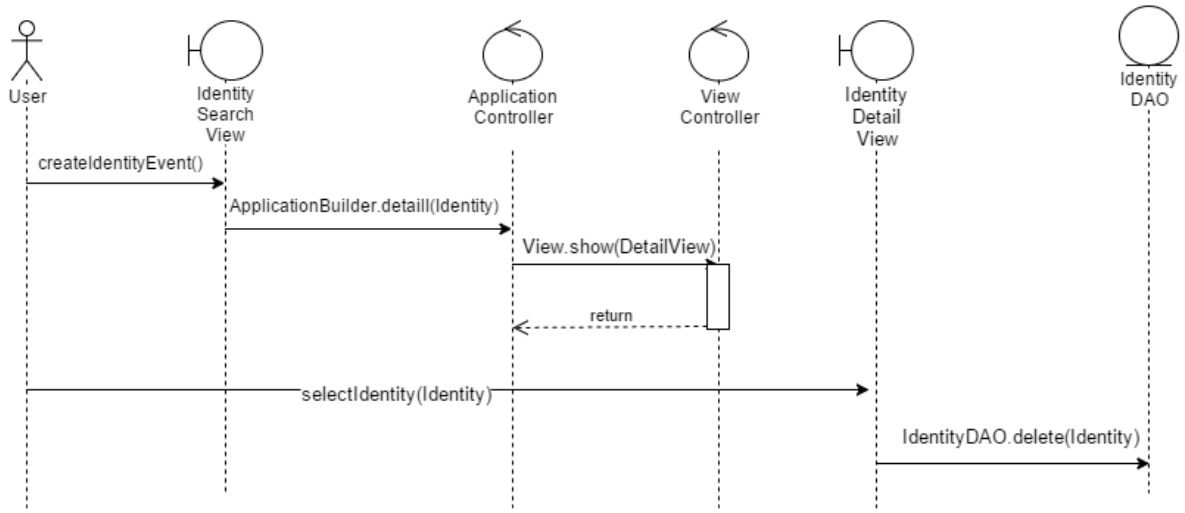
Sequence diagram Login



Sequence diagram Create



Sequence diagram Modify



Sequence diagram Delete

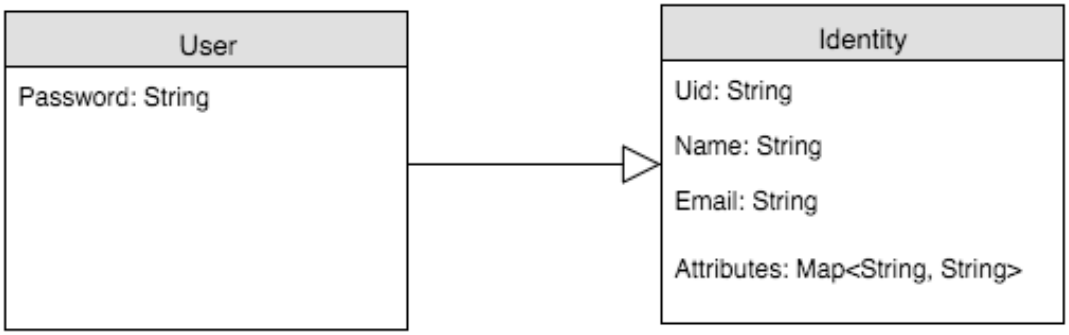
4. Conception

4.1. Algorithm implementation

Search operation:

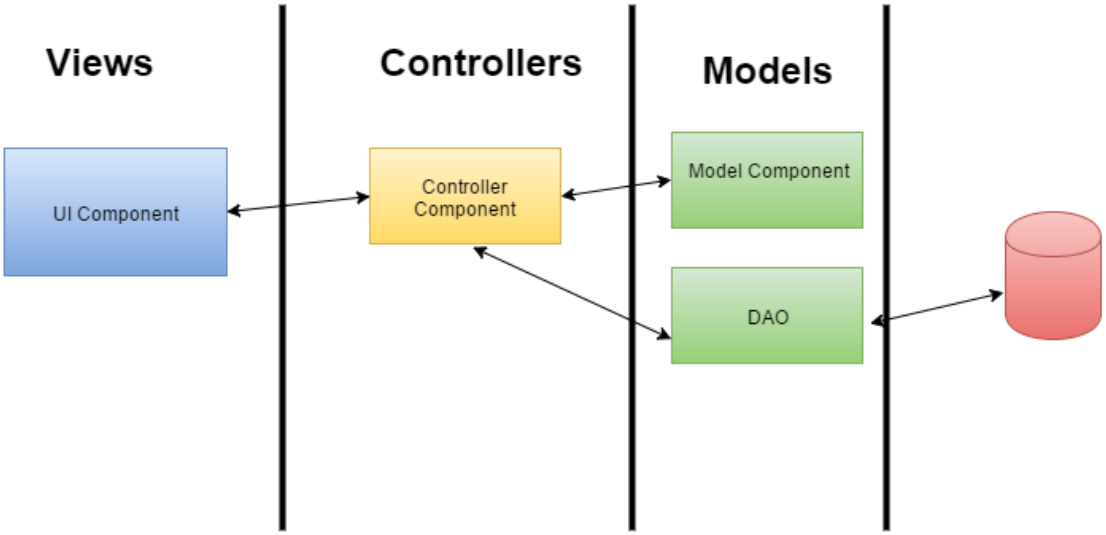
- With XML: we follow the simplest way to search an identity is to go through the XML file until we find what we need based on some criteria. This way is not effective.
- With Relational Database: thanks to SQL, search operation is provided with easiness and efficiency in the search engine.
- To make it more effective, we can cache the result of list and search in the memory (if possible) and limit read/write to files and DB.
- Sort operation (not implemented):
- With XML: We read all identities and then we can use some sorting algorithms (merge sort, quick sort).
- Usage of reflection for handling the XML files.
- With RDB: We can easily implement sort operation with SQL.

4.2. Data structures:



4.3. Global application flow

The following diagram represents the interaction of the application components:



4.4. Global schema and major features schema

5. Desktop application description

5.1. Login view

This view is the initial view of the application. It forces the admin to provide the name and password to login. If login is successful, the admin will go to the menu view.

On this view, admin can choose the type of the storage to read and write the data (XML or Derby).

5.2. Menu view

This view shows the menu of identity operation options including identities listing and identity creating as well as the logout option. When admin chooses to logout, he will be redirected to the login view.

5.3. List and search view

This view allows the admin to view all the identities as well as to search for an identity with specific criteria. It will display some information of identities in a list like table. The admin can click on an identity to view the detail of it in the detail view.

5.4. Detail view

This view allows the admin to view the detail information of an identity. He can modify the main data (name, email) as well as the attributes (add, delete, modify). And he can delete the identity with confirmation.

After deleting operation or explicitly hitting a button, the view will be changed back to the list view or the admin can logout of the application.

5.5. Create view

This view allows the admin to create a new identity with a flexible number of attributes. He can go to list view or logout from here.

6. Configuration and installation

6.1. Pre-requisites and installation

- Download & install Java SDK, Eclipse, and Database Management Application (if needed).

6.2. Configuration

6.2.1. Data storage preparation

With XML: Create the following XML file, named users.xml

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<users>
    <user>
        <property name = "displayName">admin</property>
        <property name = "password">Epita</property>
    </user>
</users>
```

With DB:

- Should have enough disk space.
- Create database.

```
create table identities (id integer not null
    generated always as identity(start with 1, increment by 1),
displayName varchar(128) not null,
email varchar(128) not null,
uid varchar(128) not null,
password varchar(128),
attributes varchar(1028));
```

- Create user.
- Grant necessary privileges of the database to user.

6.2.2. Configure the config file

- Open the file “data_config.yml” in a text editor:
- Admin: Change the default admin credentials like:

admin:

name: admin

password: password

If this is left blank, an admin user will be created automatically with credential “admin/password”

+ Database: Change the credential of DB like:

database:

type: derby

username: username

password: password

The following image is an example:

```
admin:
    name: admin
    password: password
database:
    adapter: derby
    dbname: iamdb
    username: user
    password: user
```

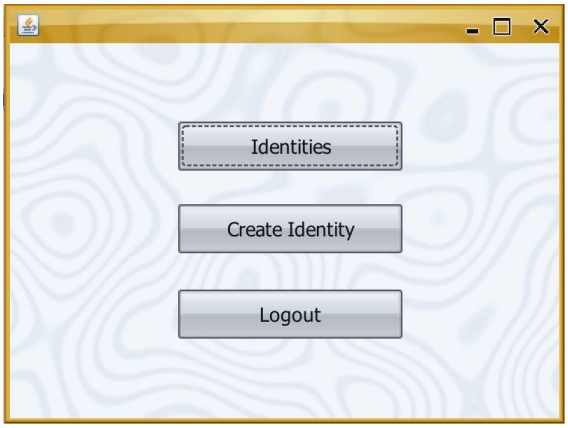
data_config.yml

There could be some more configuration for database like host, port but for this version we keep just type, username and password to make it simple.

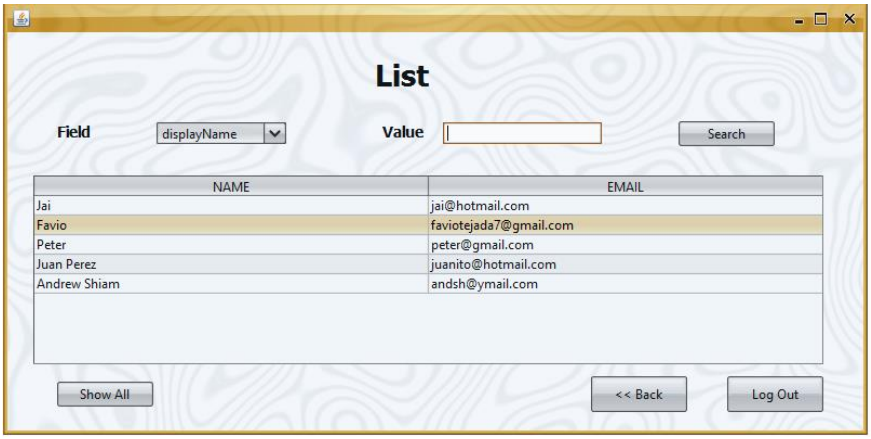
7. IAM application screenshots

Login screenshot: admin should provide username and password to login with an option of data storage to continue or quit the application.

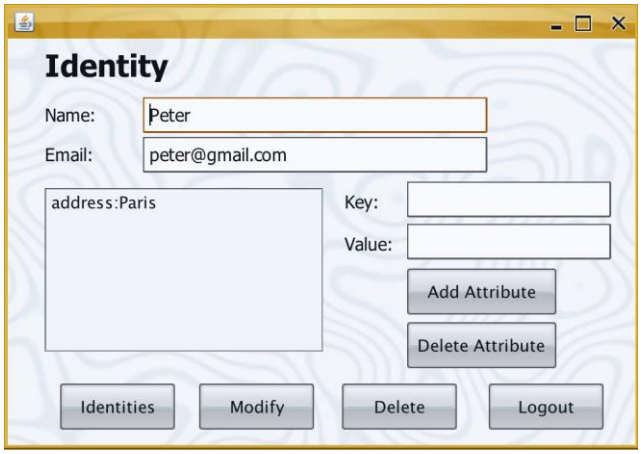
Menu screenshot: A list of actions on menu screen such as "view the identities", "create a new identity" or "logout".



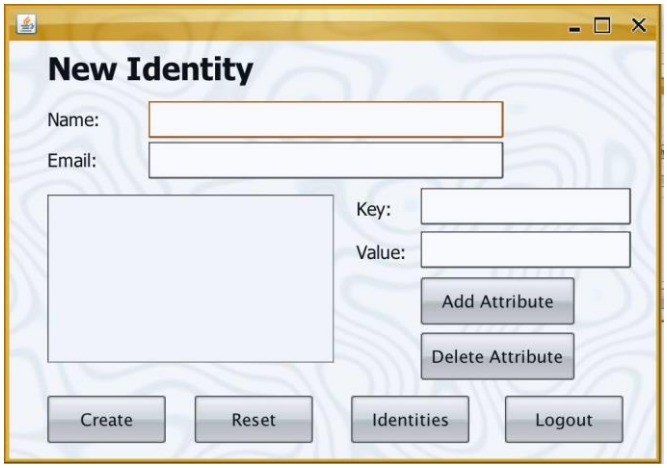
List screenshot: Display all the identities in a table and allow admin to search with displayName and email.



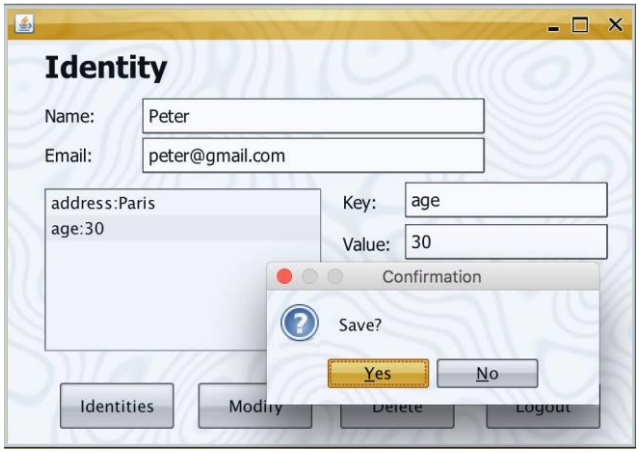
Detail screenshot: Show the identity information in details. Admin can modify or delete this identity.



Create screenshot: Create a new identity



Modify screenshot: Confirmation of modifying an identity



8. Bibliography

https://www.youtube.com/watch?v=aKbSlgi5h68&index=1&list=PLtTVgBdymZBjtal6O9bw1esdLFiF73hdz

- http://docs.oracle.com/javase/7/docs/technotes/guides/jdbc/
- https://github.com/stleary/JSON-java
- https://code.google.com/archive/p/snakeyaml/
- https://db.apache.org/derby/
- http://www.w3schools.com/xml/
- http://insubstantial.github.io/insubstantial/substance/
- https://eclipse.org/windowbuilder/